Accompanying notes to the presentation and session

Compiled by:

Bertie Buitendag &
Hannie van der Merwe

Disclaimer: This set of notes is presented under ULA terms (Use it, Lose it, or Abuse it) conditions and is presented as is. 😉

Contents

Foreword

The following example excersise contains content covering some of the South African Curriclumlum and Assessment Policy for Information Technolgy curriculum in high school (Grades 10 to 12).

The topics covered include:

Basic coding, arrays and OOP principles also features as part of the exercise.

Look out for the following pictures to indicate –

the code to achieve the required outcome : 

the function of the code (the question) : 

The **incomplete** Delphi program **QBDriveSchoolExample_p.dproj** in the **DB_Example** folder can be used to code the exercise.

The **completed** / **solution** Delphi program is included in the **DB_Example - Solution** folder to execute if you want to see what the program is suppose to do.

Background - Quick Banana Driving School

SCENARIO

Learning to drive, and to drive safely is an important skill especially in South Africa. Quick Banana Driving school offers the answer. The database contains data relating to the schools learners, their instructors as well as the lesson bookings that the learners have made for driving lessons.

The design of the database is given below:

The Database QuickBanana.accdb comprise of four tables:

Drivebookings table which contain records of all the drivers lessons.
Instructors table, which contain records relating to the instructors.
LearnerClients table, which contain records relating to the learner clients.
LearnersQuestionBank table, which contain the sample test question bank and answers.

Excerpts of each of the table records are given below as well as the data dictionary entry of the table:

Drivebookings Table:

DriveBookings

Cnt	InstructorNr	LearnerClie	LessonDate	LessonTime	LessonDurat
10036	INS007	BL751494	2012/08/08	16:00	1
10037	INS005	BL741044	2012/08/08	11:00	1
10038	INS001	BL541157	2012/08/09	10:30	2
10039	INS001	BL811079	2012/08/10	15:00	1
10040	INS002	BL241432	2012/08/08	16:00	1
10041	INS002	BL741472	2012/08/08	07:00	1
10042	INS002	BL331141	2012/08/08	10:00	2
10043	INS002	BL511031	2012/08/08	16:00	1
10044	INS002	BL721390	2012/08/08	15:00	

Table Design

DriveBookings

Field Name	Data Type
Cnt	AutoNumber
InstructorNr	Text
LearnerClientNr	Text
LessonDate	Date/Time
LessonTime	Date/Time
LessonDuration	Number

Instructors Table

Instructors

InstructorPer:	InstructorSAII	InstructorNa	InstructorSu	InstructorCc	InstructorLic	EmpBasicSa
INS001	5510205643086	Bongi	Nel	0832213456	EB,C1,A	R 162 900.00
INS002	6002104456083	CJ	Mahlangu	0795650023	EB,A1,C1,EC1	R 167 900.00
INS003	7710105339086	Clifford	Nkadibeng	0722215328	EB,C1,EC1,A	R 167 900.00
INS004	7509203776086	Jackson	Tshotshana	0826657894	EB,B,C1,C	R 167 900.00
INS005	8001234553089	Lerato	Mangumeng	0847765510	EB,C1,A,EC1	R 162 900.00
INS006	7004245578088	Neels	Baloyi	0797538810	EB,C1,C	R 182 900.00
INS007	8106065221084	Samson	de Bruin	0825547890	EB,EC1,A,C1	R 167 900.00
INS008	6709144500086	Sue	Makamela	0836778080	EB,A,EC1	R 162 900.00
INS009	7208105667085	Suzette	Lakuma	0726675091	EB,C,C1	R 182 900.00
INS010	7802164300081	Temperance	November	0838081198	EB,EC,EC1	R 167 900.0

The InstructorLicenceCodes field contain the values to represent the different driving licence codes the instructor is allowed to teach e.g.

EB – Standard Vehicle with a trailer or caravan
A – Motorcycle
C – Small truck
Etc.

Table Design

Instructors

Field Name	Data Type
InstructorPersID	Text
InstructorSAID	Text
InstructorName	Text
InstructorSurname	Text
InstructorContactNr	Text
InstructorLicenceCodes	Text
EmpBasicSalPA	Currency

LearnerClients Table

LearnerClients

LearnerClientN	LearnerName	LearnerSurname	LearnerID	Learne	LearnerCell	Learneri	PreferredIn	LearnersLicenceCo	LearnersLicenc
BL101024	Justine	Claasen	7207213461037	Miss	0822595794	B	INS004	8084UI_ZA81	2014/03/12
BL101080	Athina	Jason	7904044203085	Miss	0727054116	B	INS004	0118TH_ZA59	2014/03/13
BL101111	Shameez	Potgieter	7706114404075	Miss	0722365709	B	INS004	1111HQ_ZA19	2012/09/24
BL101297	Lee	Forson	7804072102007	Miss	0737183829	B	INS004	1830AP_ZA80	2013/09/23
BL101325	Clint	Steineveldt	7705135564050	Mr	0738816226	EC1	INS005	8271OU_ZA93	2013/02/03
BL101355	Darryl	Van Den Berg	8108116877012	Mr	0735415672	B	INS004	6285YT_ZA21	2013/10/01
BL101407	Lauren	Hagen	9102262552015	Miss	0842595794	A	INS005	7179PP_ZA52	2013/01/23
BL111073	Mthokozisi	Nondzimba	7710175703045	Mr	0724146572	A1	INS002	9784SJ_ZA89	2013/09/29
BL111216	Noxolo	Nase	7507141418019	Miss	0732595794	B	INS004	6942MT_ZA60	2013/03/19
BL111332	Bernita	Camons	6806023282074	Miss	0737256713	C1	INS003	6877BT_ZA46	2013/09/10
BL111382	Jarred	Wicombe	7510275788035	Mr	0849858126	C1	INS009	2445KM_ZA41	2014/03/10
BL111397	Dustin	Carney	7912188744079	Mr	0848816226	C1	INS005	3696SK_ZA72	2013/05/

Table Design

LearnerClients

Field Name	Data Type
LearnerClientNr	Text
LearnerName	Text
LearnerSurname	Text
LearnerID	Text
LearnerTitle	Text
LearnerCell	Text
LearnerDriverCode	Text
PreferredInstructor	Text
LearnersLicenceCode	Text
LearnersLicenceExpirydate	Date/Time

LearnersQuestionBank Table

LearnersQuestionBank

QuestionNr	Question	Option1	Option2	Option3	Option4	Answer	PicRef
Q1	"IF YOU WANT	Switch on your	"Give the nece	Apply the brak	None of the ab	3	None
Q10	THE FOLLOWIN	People per axe	Two tyres nee	Axel mass load	Maximum veh	3	06.bmp
Q11	THE FOLLOWIN	Not a good pla	Hitch-hiking pr	Possible highja	No freeloaders	2	07.bmp
Q12	THE FOLLOWIN	You can only se	Hawkers prohi	No selling allo	No cannon bal	2	08.bmp
Q13	THE FOLLOWIN	Night conditio	No clouds at ni	Full moon and	Twinkle twinkl	1	09.bmp
Q14	THE FOLLOWIN	Hawkers may s	Business hours	Reserved for s	Two periods or	4	10.bmp
Q15	THE FOLLOWIN	Holiday resort	Camping area	Woonerf	Children playir	3	11.bmp
Q16	THE FOLLOWIN	Motor cycle en	Bus passenger	some sort of v	126 cc will not	1	12.bmp
Q17	THE FOLLOWIN	Minimum Spee	Maximum spe	Maximum num	Average load p	3	13.bmp
Q18	THE FOLLOWIN	Person at mov	King Arthur ge	Man struggling	Construction w	4	14.bmp

Table Design

LearnersQuestionBank

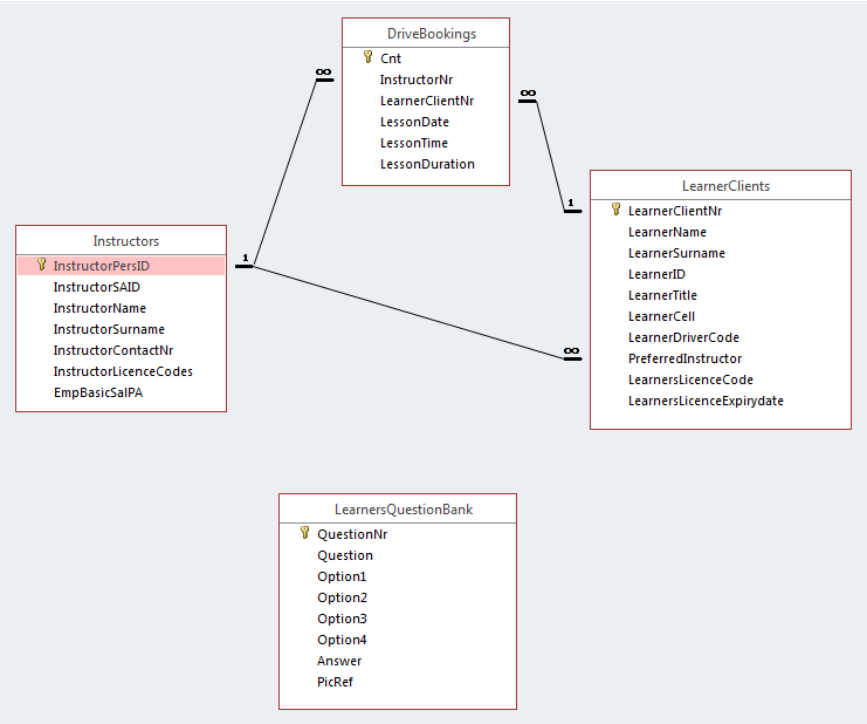
Field Name	Data Type
QuestionNr	Text
Question	Text
Option1	Text
Option2	Text
Option3	Text
Option4	Text
Answer	Number
PicRef	Text

TAKE note: The Cnt field is the primary key of the DriveBookings table and is managed by the DBMS. (Database Management Software / DB Engine)

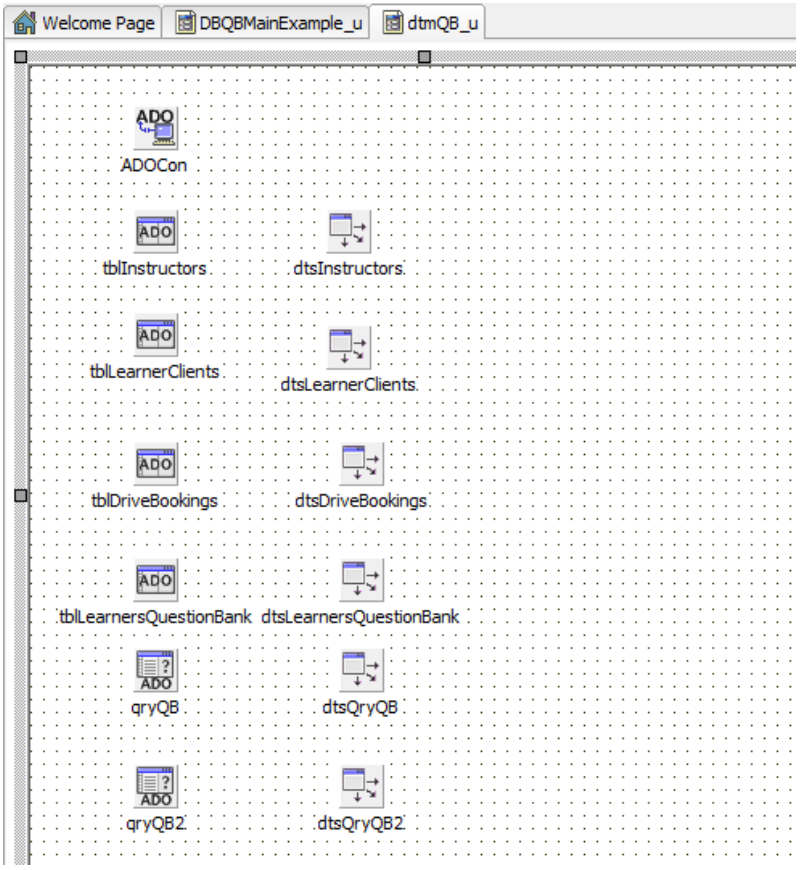
The duration of a lesson may be 1 or 2 hours

The InstructorNr and the LearnerClientNr fields are foreign key fields to the Instructors and LearnerClients tables respectively.
One instructor may book one or more drives and may have one or more learner clients.
One learner client may book one or more drives.

The diagram below provides an overview of the entities and the relationships between them.



The following components are included as part of the datamodule

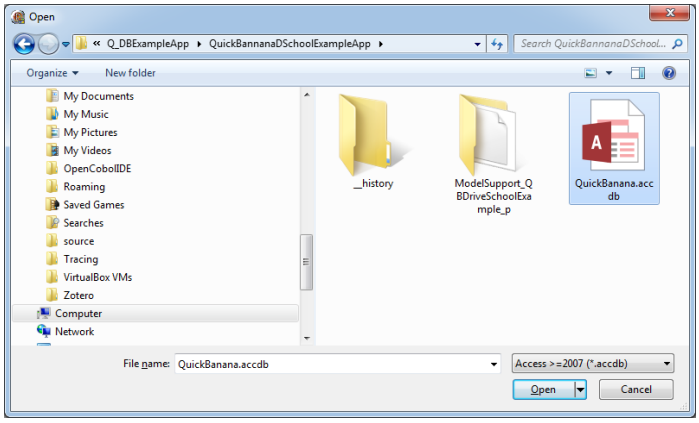


For every table as part of the DB a corresponding ADO table component and applicable datasource component is included.

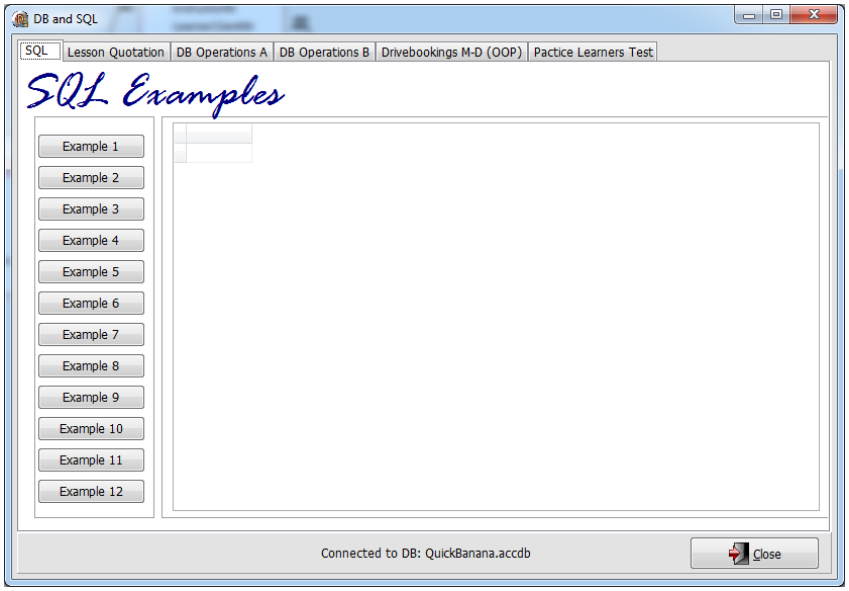
Two query and related datasource components are also included as part of the Datamodule (DM) named **dtmQBDB**

When the application is run for the first time the following screen appears.

The user is then allowed to select and connect to a DB, you have the option to connect to an mdb or accdb access database.



When the user clicks on the Open button the following interface is shown



The SQL Tabsheet

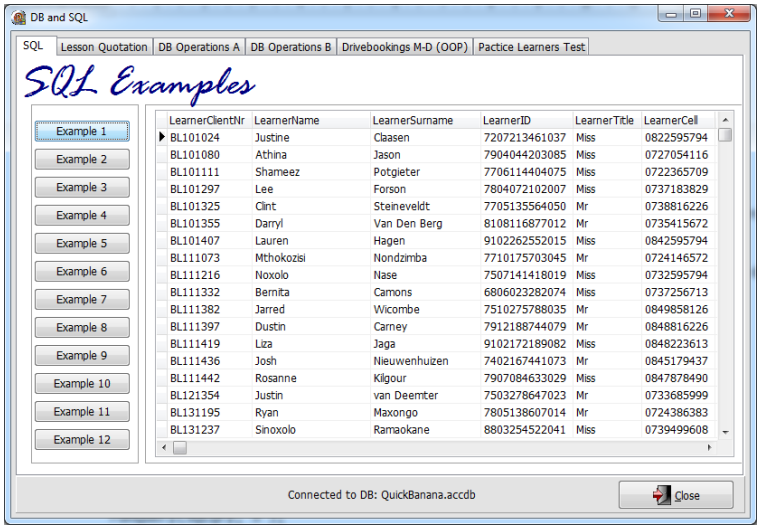
The 12 different buttons placed on the tabsheet each performs a different query to the database and the results where applicable are shown in the dbgrdSQLResult DBGrid.

Each of the queries executed and their code is briefly discussed next.

Example 1 – simple select

```
procedure TfrmQBDBExample.btnEx1Click(Sender: TObject);
begin
//  Type your query between the '' like the example below
SQLQry := 'Select * From LearnerClients '; // Query text  between the ''

// Don't change code below
runQuery(SQLQry,'S');
end;
```

This query lists all the records and all the columns (fields) from the LearnerClients table.



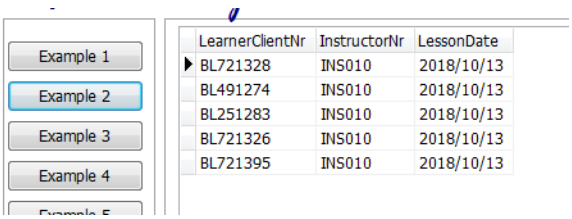
Example 2 – select with criteria ("filter")

```
procedure TfrmQBDBExample.btnEx2Click(Sender: TObject);
begin

SQLQry := 'Select LearnerClientNr ,InstructorNr, LessonDate  ';
SQLQry := SQLQry + ' From DriveBookings where Lessondate = #2018/10/13# ';
SQLQry := SQLQry + ' and InstructorNr = "INS010"  ';

// Don't change code below
runQuery(SQLQry,'S');
end;
```

The Lessondate field is declared as a DateTime field,we place date values between two #s

We place string literals between two "s

This query lists all the LearnerClientNr, the InstructorNr and the LessonDate for the instructor with the InstructorNr INS010 for 2018/10/13

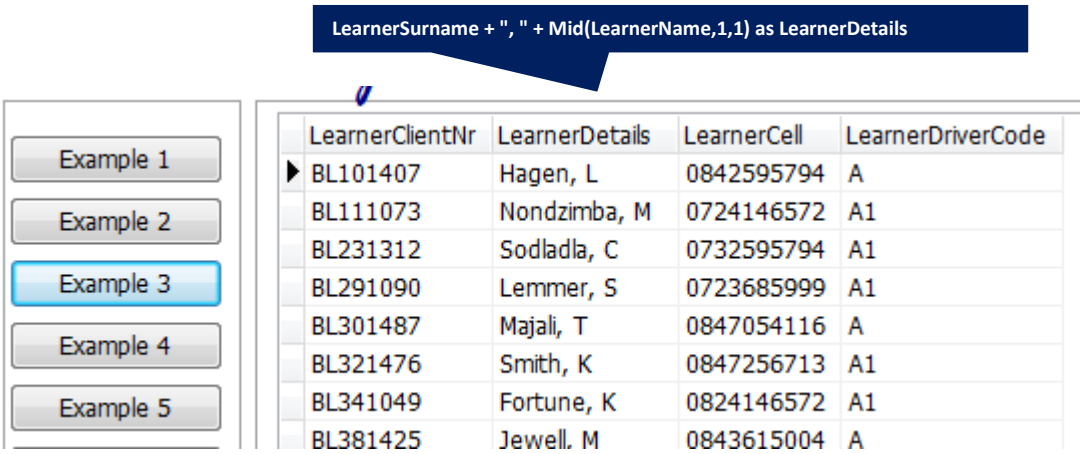


Example 3 – select with more than one criteria

```
SQLQry := ' Select LearnerClientNr,  ';
SQLQry := SQLQry + ' LearnerSurname + ", " + Mid(LearnerName,1,1) as LearnerDetails,';
SQLQry := SQLQry + ' LearnerCell, LearnerDriverCode ';
SQLQry := SQLQry + ' From LearnerClients ';
SQLQry := SQLQry + ' Where LearnerTitle <> "Mr" and ';
SQLQry := SQLQry + ' LearnerDriverCode = "A" or LearnerDriverCode = "A1" ';
// Don't change code below
runQuery(SQLQry,'S');
end;
```

This query lists the LearnerClientNr, the Surname and First letter of the name of the learner with a , in-between. This field is listed with the column heading LearnerDetails. The Cell and LearnerDriverCode is also listed. Only records for Male learnerclients which is learning for A or A1 drivers licences are listed.

Note: The Mid function returns one character from the first letter of the LearnerName onwards.



Example (extra)

The following SQL example was executed using the DelphiQueryBuilder application.

```
Select LearnerName,  Left(LearnerName,4) as Left4,
Right(LearnerName, 3) as Right3, Mid(LearnerName,3,2) as Mid32
From LearnerClients
```

Results in

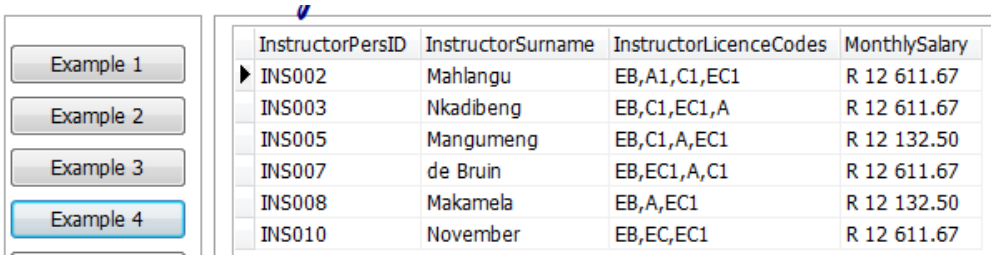
Table View | Query Result | Table Structure

LearnerName	Left4	Right3	Mid32
Tiisetso	Tiis	tso	is
Maxine	Maxi	ine	xi
Chad	Chad	had	ad
Fabian	Fabi	ian	bi
Anysha	Anys	sha	ys
Kavish	Kavi	ish	vi
Rehana	Reha	ana	ha
Nthabeleng	Ntha	eng	ha
Dean	Dean	ean	an
Stuart	Stua	art	ua

Example 4 – select with criteria and formatted calculated field

```
SQLQry := ' Select InstructorPersID, InstructorSurname, InstructorLicenceCodes, ';
SQLQry := SQLQry + ' FORMAT(((EmpBasicSalPA / 12) + (EmpBasicSalPA * 0.0125)),"Currency") as
MonthlySalary ';
SQLQry := SQLQry + ' FROM Instructors  ';
SQLQry := SQLQry + ' Where InstructorLicenceCodes Like "%EC1%" ';
```

This query lists some fields from the Instructors table and includes a calculated column named MonthlySalary formatted as a currency field which is the Basic Salary per annum (EmpBasicSalPA) divided by 12 to which 1.25% of the EmpBasicSalPA is added and returned. Only records are listed for instructors that is able to teach EC1 licences.



InstructorPersID	InstructorSurname	InstructorLicenceCodes	MonthlySalary
INS002	Mahlangu	EB,A1,C1,EC1	R 12 611.67
INS003	Nkadibeng	EB,C1,EC1,A	R 12 611.67
INS005	Mangumeng	EB,C1,A,EC1	R 12 132.50
INS007	de Bruin	EB,EC1,A,C1	R 12 611.67
INS008	Makamela	EB,A,EC1	R 12 132.50
INS010	November	EB,EC,EC1	R 12 611.67

Example (extra)

Some additional format examples

```
Select Format(0.2, "Percent") as FormatPercent,
Format(100, "Currency") as FormatCurrency,
Format(50,"Fixed") as FormatFixed
```

Results in

Table View | Query Result | Table Structure

FormatPercent	FormatCurrency	FormatFixed
20.00%	R 100.00	50.00

Example (extra)

```
Select LearnerSurname
From LearnerClients
Where LearnerSurname Like "_am%"
```

The % wildcard character represents any number of characters and the _ (underscore) any single one character. E.g. List the surnames which has a am from the 2nd character onwards. **Note:** more than one _ may be used. E.g.

```
Select LearnerSurname
From LearnerClients
Where LearnerSurname
Like "_a__m%"
```

Table View | Query Result

LearnerSurname
Makamba
Hageman
Mahomed

Results in

Table View | Query Result | Table Structure

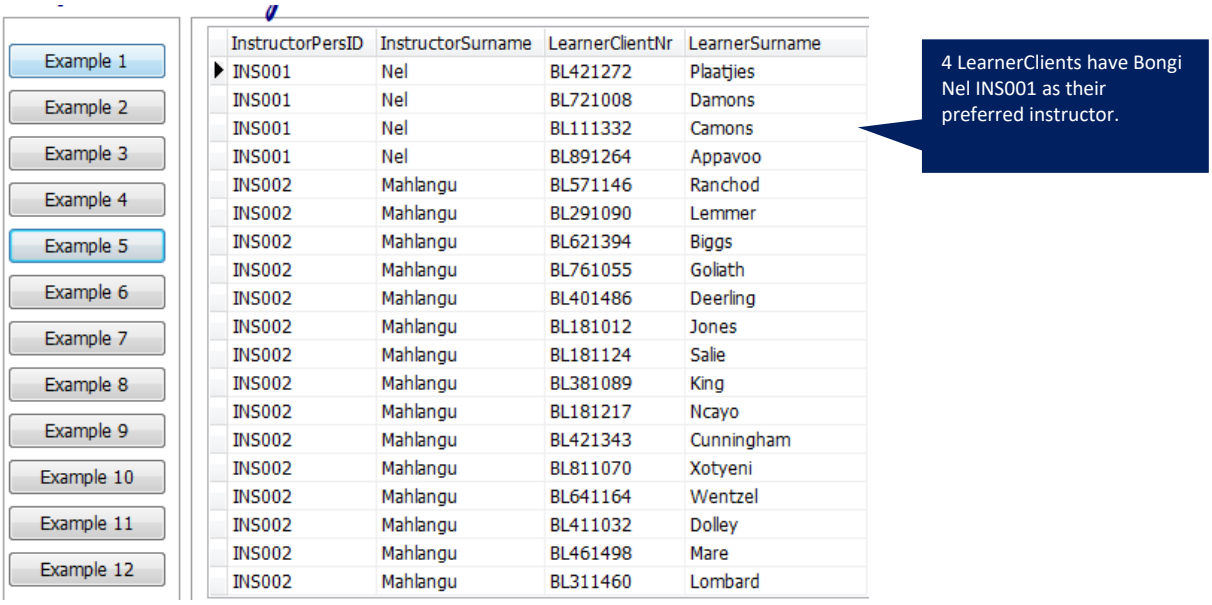
LearnerSurname
Camons
Ramaokane
Rama
Camagu
Jam
Damon
Damons

Example 5 – select from more than one table

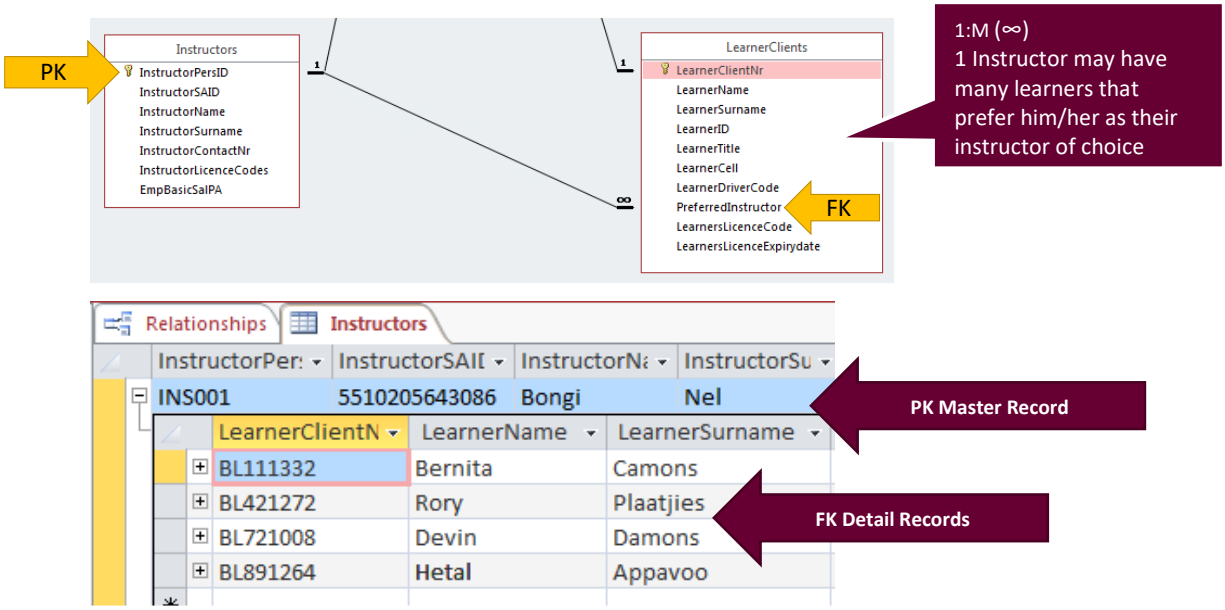
```
SQLQry := ' SELECT I.InstructorPersID, I.InstructorSurname,  ';
SQLQry := SQLQry + ' L.LearnerClientNr, L.LearnerSurname ';
SQLQry := SQLQry + ' FROM Instructors I, LearnerClients L  ';
SQLQry := SQLQry + ' WHERE I.InstructorPersID = L.PreferredInstructor ';
```

This query lists some fields from the Instructors and Learners table with an additional field to show the details of the preferred instructor for the learner. In other words, the instructor details and the learners who selected the instructor as their preferred instructor are shown.

The primary key of the instructors table is joined to the foreign key of the learners table.



InstructorPersID	InstructorSurname	LearnerClientNr	LearnerSurname
INS001	Nel	BL421272	Plaatjies
INS001	Nel	BL721008	Damons
INS001	Nel	BL111332	Camons
INS001	Nel	BL891264	Appavoo
INS002	Mahlangu	BL571146	Ranchod
INS002	Mahlangu	BL291090	Lemmer
INS002	Mahlangu	BL621394	Biggs
INS002	Mahlangu	BL761055	Goliath
INS002	Mahlangu	BL401486	Deerling
INS002	Mahlangu	BL181012	Jones
INS002	Mahlangu	BL181124	Salie
INS002	Mahlangu	BL381089	King
INS002	Mahlangu	BL181217	Ncayo
INS002	Mahlangu	BL421343	Cunningham
INS002	Mahlangu	BL811070	Xotyeni
INS002	Mahlangu	BL641164	Wentzel
INS002	Mahlangu	BL411032	Dolley
INS002	Mahlangu	BL461498	Mare
INS002	Mahlangu	BL311460	Lombard



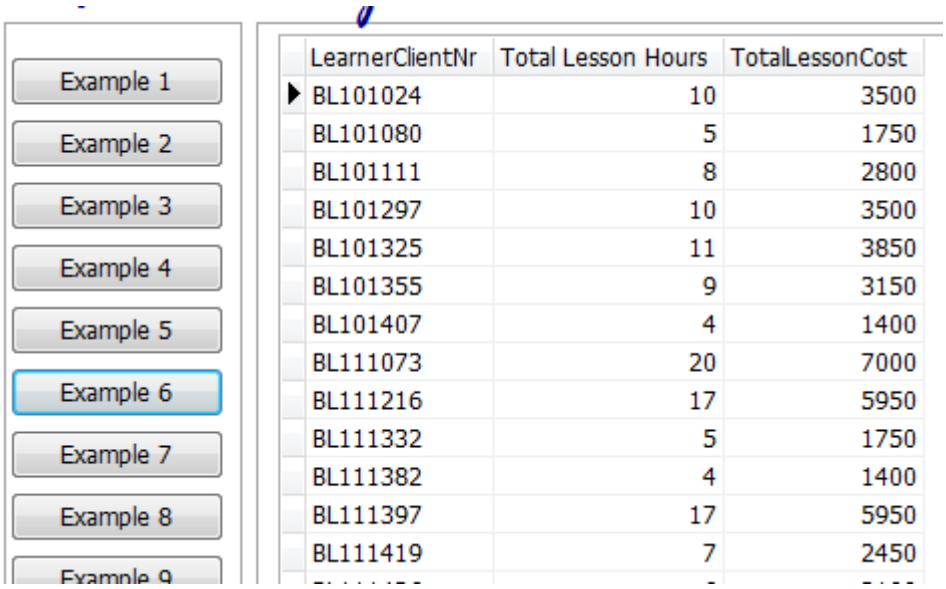
Example 6 – slect with calculated field and grouping

```
SQLQry := ' Select LearnerClientNr , Sum(LessonDuration) as [Total Lesson Hours] , ';
SQLQry := SQLQry + ' [Total Lesson Hours] * 350 as TotalLessonCost ';
SQLQry := SQLQry + ' From DriveBookings ';
SQLQry := SQLQry + ' Group by LearnerClientNr ';
```

This query determines the total lesson hours per LearnerClient as the sum of the lessonduration which is stored in the DriveBookings table. A Lesson per hour is charged at R350.00.

Note: The SQL statement includes an aggregate statement. For each additional non calculated or normal field statement returned a corresponding group by field is required.

Observe the use of the [ ] to represent column headings which contain a space. Also note how the [Total Lesson Hours] is used as a field in the calculation of the TotalLessonCost.



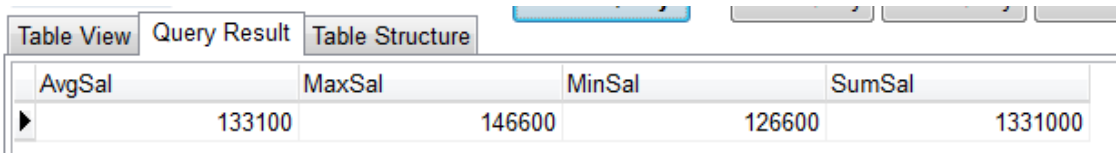
LearnerClientNr	Total Lesson Hours	TotalLessonCost
BL101024	10	3500
BL101080	5	1750
BL101111	8	2800
BL101297	10	3500
BL101325	11	3850
BL101355	9	3150
BL101407	4	1400
BL111073	20	7000
BL111216	17	5950
BL111332	5	1750
BL111382	4	1400
BL111397	17	5950
BL111419	7	2450

Example (extra)

Some other examples of aggregate functions

```
Select Avg(EmpBasicSalPA) as AvgSal,
Max(EmpBasicSalPA) as MaxSal,
Min(EmpBasicSalPA) as MinSal,
Sum(EmpBasicSalPA) as SumSal
From Instructors
```

Result of the SQL statement above



AvgSal	MaxSal	MinSal	SumSal
133100	146600	126600	1331000

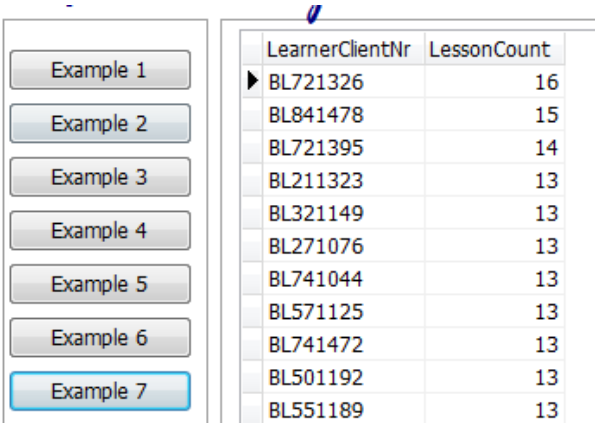
Example 7 – select with calculated field, group and order (sort)

```
SQLQry := ' Select LearnerClientNr , Count (LearnerClientNr) as LessonCount   ';
SQLQry := SQLQry + ' FROM DriveBookings ';
SQLQry := SQLQry + ' GROUP BY LearnerClientNr ';
SQLQry := SQLQry + ' ORDER BY 2 desc ';
```

This query determines the number of lessons per LearnerClient. The Count aggregate function is returned and due to the fact that another non calculated field (i.e. a field from the table is retured in conjunction with the aggregate the GROUP BY clause is required.

Count would normally return and count all the records, but this should be done per LearnerClientNr, therefore the use of the Group by.

The records that is to be retuned should also be sorted in descending sequence according to the 2nd column and therefor the use of the Order by 2 desc. The 2 refer to the second column.



LearnerClientNr	LessonCount
BL721326	16
BL841478	15
BL721395	14
BL211323	13
BL321149	13
BL271076	13
BL741044	13
BL571125	13
BL741472	13
BL501192	13
BL551189	13

Example 8 – select with sub query

```
SQLQry := ' Select Count(*) as SalLessThanAveCount  ';
SQLQry := SQLQry + ' FROM Instructors  ';
SQLQry := SQLQry + ' WHERE EmpBasicSalPA <  ';
SQLQry := SQLQry + ' (Select AVG(EmpBasicSalPA) From Instructors)  ';
```

The subquery is executed first and returns the average salary.

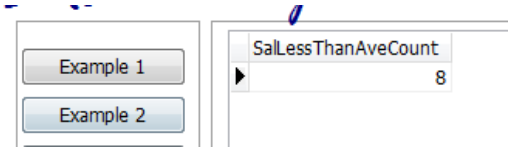
NOTE: Count is also an aggregate function

This query returns the number of instructors that earns less than the average salary for all the instructors.

This query implements a sub query. The sub query is executed first and the value that is returned is used as a condition for the where clause of the main query.

First the Average Salary is returned, then all the records are counted where the EmpBasicSalPA is less then the average salary.

The query shows that there is 8 employees that earn less than the average salary for all employees.



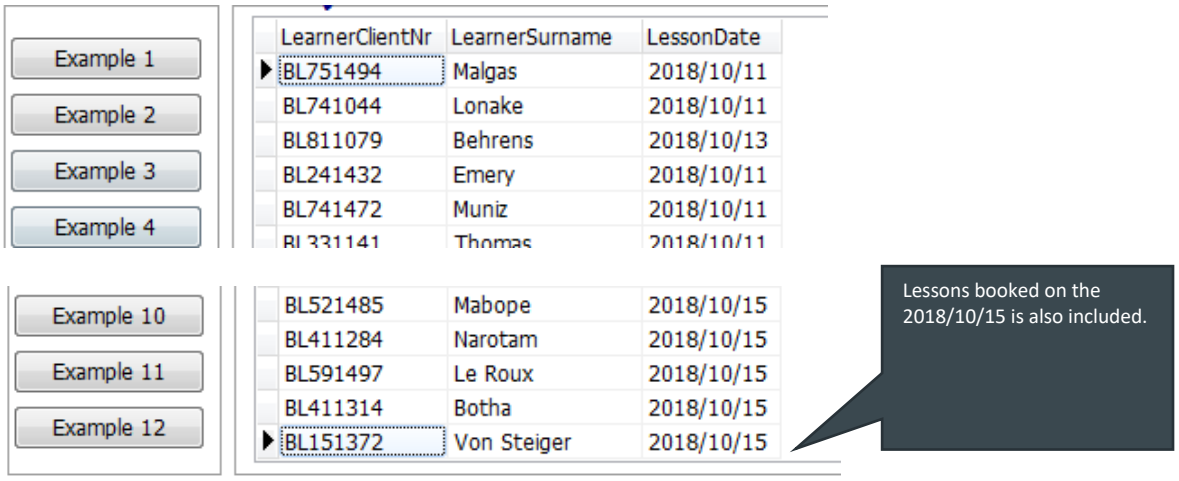
SalLessThanAveCount
8

Example 9 – select from two tables with date criteria

```
SQLQry := ' Select B.LearnerClientNr, L.LearnerSurname, B.LessonDate ';
SQLQry := SQLQry + ' From DriveBookings B, LearnerClients L ';
SQLQry := SQLQry + ' Where B.LearnerClientNr = L.LearnerClientNr ';
SQLQry := SQLQry + ' and B.LessonDate Between #2018/10/01# and #2018/10/15# ';
```

This query lists the LearnerClientNr, the LearnerSurname and the Lesson dates booked by the learner. Two PK field from the LearnerClients table is joined with the corresponding FK field of the Drivebookings table. Only records lessons booked between the 2018/10/01 and 2018/10/15 are shown.

Take note: The BETWEEN operator are inclusive, which means that for our example 2018/10/01 and 2018/10/15 is included in the comparison and lessons booked on these dates are also returned.

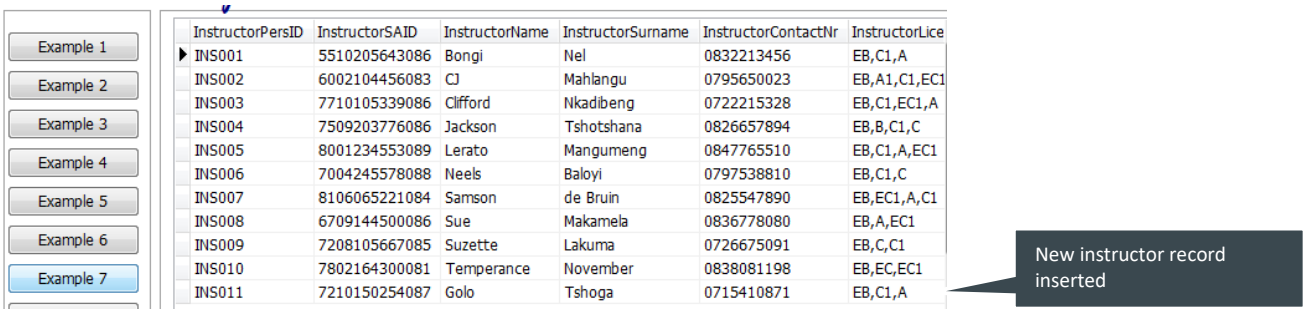


Example 10 – insert into

```
SQLQry := ' INSERT INTO Instructors   ';
SQLQry :=  SQLQry + ' Values
("INS011","7210150254087","Golo","Tshoga","0715410871","EB,C1,A",120000)';
```

This query implements a DML statement (Data Manipulation Language) to modify/manipulate the data in a table or set of tables.

For this example a new records is to be added to the Instructor table. Take note of how the values, i.e. the field values are inserted.



Example 11 - update

```
SQLQry := 'UPDATE Drivebookings SET InstructorNr = "INS002" ';
SQLQry := SQLQry + ' WHERE InstructorNr = "INS010" and LessonDate >= Date() ';
```

This query implements a DML statement (UPDATE) to modify certain records. All Lessons booked for instructor INS002 should be changed to INS010 from tomorrow onwards.

The Date() function returns the current date.

Example (extra)

Additional example of the update statement

Table View | Query Result | Table Structure

Cnt	InstructorNr	LearnerClientNr	LessonDate	LessonTime	LessonDuration
10036	INS007	BL751494	2018/10/11	1899/12/30 04:00:00 F	1
10037	INS005	BL741044	2018/10/11	1899/12/30 11:00:00 A	1
10039	INS001	BL811079	2018/10/13	1899/12/30 03:00:00 F	1

Record before

DML statement

```
Update DriveBookings
Set LessonDuration = 2
Where Cnt = 10036
```

Table View | Query Result | Table Structure

Cnt	InstructorNr	LearnerClientNr	LessonDate	LessonTime	LessonDuration
10036	INS007	BL751494	2018/10/11	1899/12/30 04:00:00 F	2
10037	INS005	BL741044	2018/10/11	1899/12/30 11:00:00 A	1

Record after

Example (extra)

DBase: QuickBanana.accdb (Multiple select & drag fields) **Enter your Query Here** (Parameter names: "A" to "Z" - all CAPS)

DriveBookings	DR
Instructors	IN
LearnerClients	LE
LearnersQuestionBank	LF

Cnt
InstructorNr
LearnerClientNr
LessonDate
LessonTime
LessonDuration

```
Select * From DriveBookings
Order by LearnerClientNr
```

This query selects all booking records sorted according to the learnerclientnr. Therefore all booking records will be "grouped" accordingly.

10 | Font Size | Exec. Query | Save Query | Load Query | Clear Memo | Beg. Tr

Table View | Query Result | Table Structure

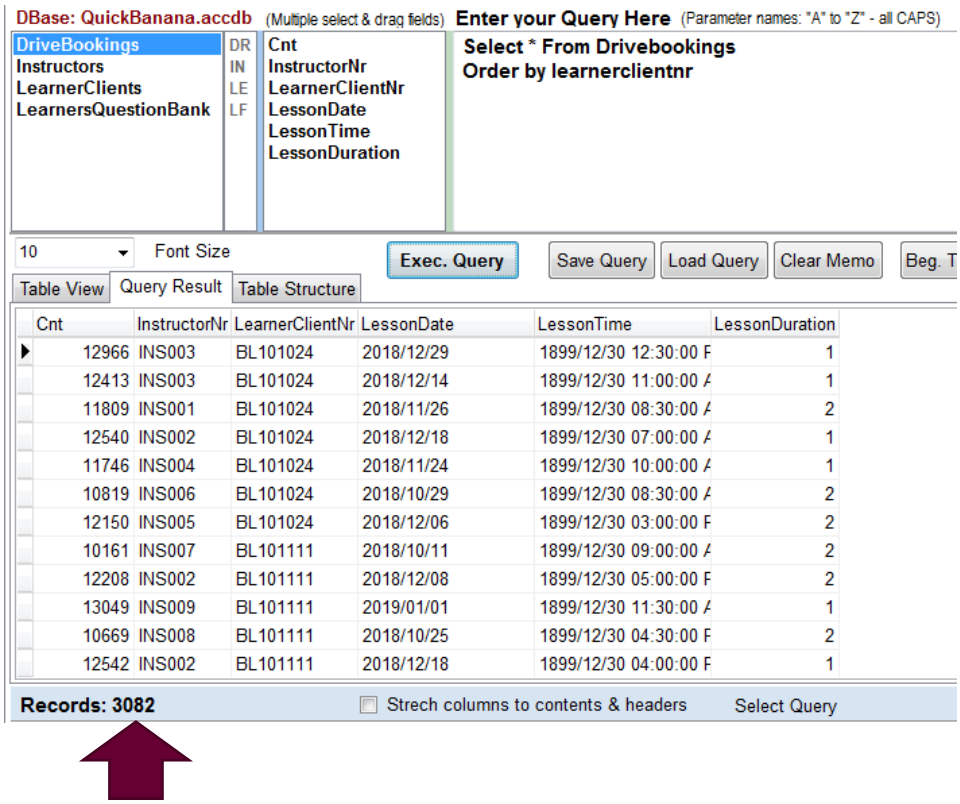
Cnt	InstructorNr	LearnerClientNr	LessonDate	LessonTime	LessonDuration
12150	INS005	BL101024	2018/12/06	1899/12/30 03:00:00 F	2
10819	INS006	BL101024	2018/10/29	1899/12/30 08:30:00 A	2
11809	INS001	BL101024	2018/11/26	1899/12/30 08:30:00 A	2
11746	INS004	BL101024	2018/11/24	1899/12/30 10:00:00 A	1
12540	INS002	BL101024	2018/12/18	1899/12/30 07:00:00 A	1
12413	INS003	BL101024	2018/12/14	1899/12/30 11:00:00 A	1
12966	INS003	BL101024	2018/12/29	1899/12/30 12:30:00 F	1
10487	INS010	BL101080	2018/10/21	1899/12/30 04:00:00 F	2
10404	INS007	BL101080	2018/10/19	1899/12/30 02:30:00 F	2
12985	INS008	BL101080	2018/12/30	1899/12/30 09:30:00 A	1
10669	INS008	BL101111	2018/10/25	1899/12/30 04:30:00 F	2
13049	INS009	BL101111	2019/01/01	1899/12/30 11:30:00 A	1

Records: 3085 | Strech columns to contents & headers | Select Query

LearnerClient BL101080 has made 3 bookings

Delete * From DriveBookings
Where LearnerClientNr = "BL101080"

After the DML statement has been executed the records have been deleted.



Example 12 – select with parameters

The full code used as part of the onClick event of this example is needs to be discussed. In this example a query is used to return the drivebooking records for a particular instructor on a particular day.

The user will enter the name of the driver and the date in two separate inputboxses. The date of the lesson and the name of the instructor is saved in two local variables named sDatum and sInsName respectively.

The query implements a join on two tables i.e. the Instructors and Drivebookings table. The InstructorPersID PK is joined with the InstructorNr FK from the drivebookings table.

Two parameters (i.e. variables used as part of a query is implemented). Take note, we indicate parameters (i.e. variables as part of a query) by using the **:** sign.

Before the query can be executed the values of the parameters should first be supplied, otherwise the SQL engine would not know what values to use.

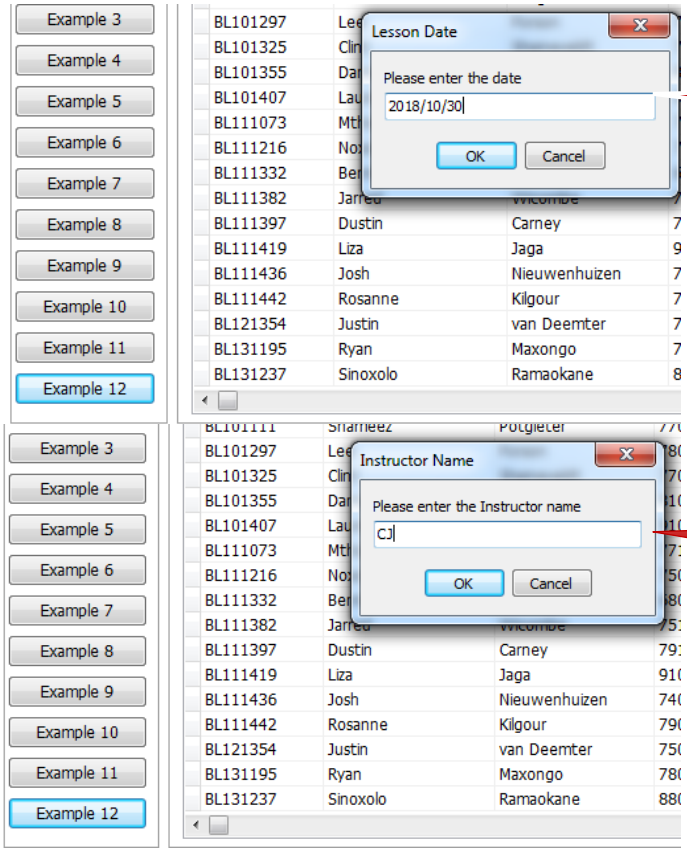
```
//-----------------------------------------------------
procedure TfrmQBDBExample.btnEx12Click(Sender: TObject);
var
  sDatum , sInsName : String;
begin
  sDatum := InputBox('Lesson Date','Please enter the date','');
  sInsName := InputBox('Instructor Name','Please enter the Instructor name','');

  SQLQry := 'Select DB.InstructorNr,I.InstructorName, DB.LearnerClientNr, DB.LessonDate ';
```

```
  SQLQry := SQLQry + ' From Instructors I, Drivebookings DB ';
  SQLQry := SQLQry + ' Where I.InstructorPersID = DB.InstructorNr and ';
  SQLQry := SQLQry + ' DB.LessonDate = :Datum ';
  SQLQry := SQLQry + ' and I.InstructorName = :InsName ';

  with dtmQBDB.qryQB do
   begin
     Active := false;
     SQL.Clear;
     SQL.Add(SQLQry);
     Parameters.ParamByName('Datum').Value := sDatum;
     Parameters.ParamByName('InsName').Value := sInsName;
     Active := true;
   end;

   SetGridColumnWidths(dbgrdSQLResult);

end;
//-----------------------------------------------------
```

:Datum and :InsName are two variables as part of the query. Variables in a query is referred to as paramaters

Before the query could be run the paramters first need to be assigned some values.

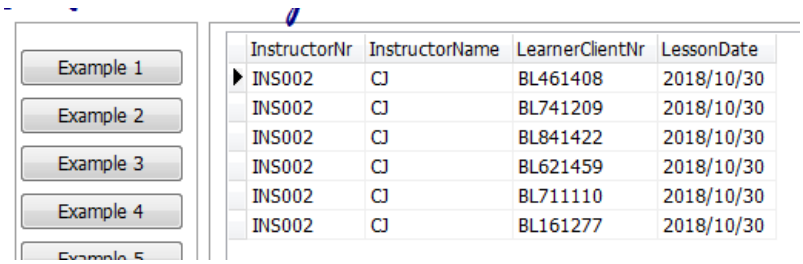
Clicking on the button



Supplying the Lesson Date

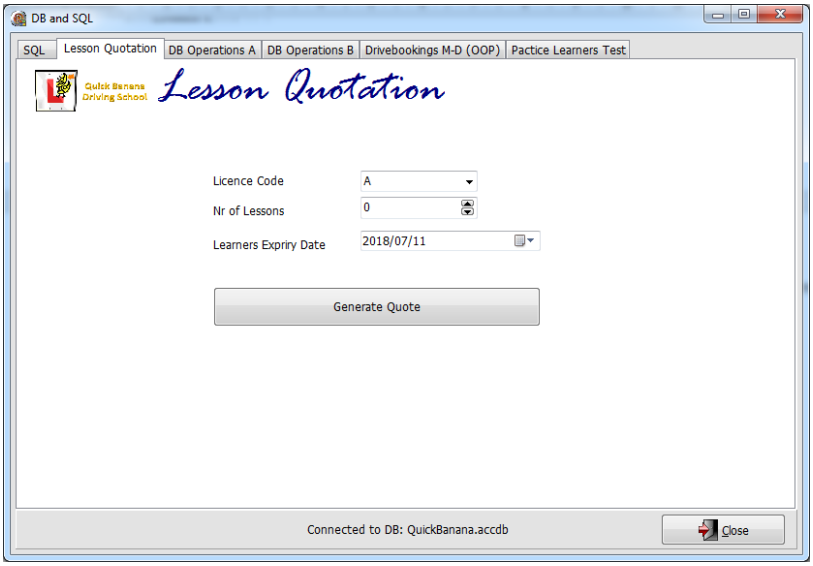
Supplying the Instructor name

Records returned

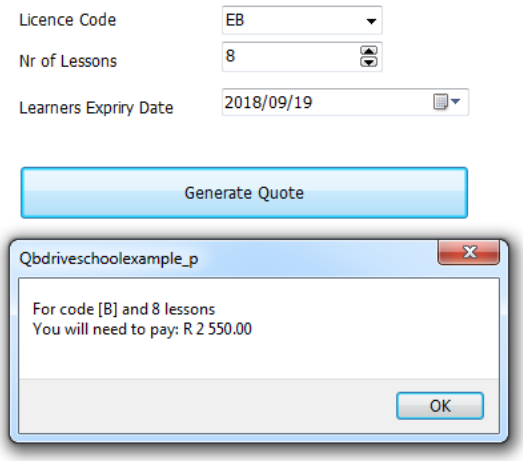


InstructorNr	InstructorName	LearnerClientNr	LessonDate
INS002	CJ	BL461408	2018/10/30
INS002	CJ	BL741209	2018/10/30
INS002	CJ	BL841422	2018/10/30
INS002	CJ	BL621459	2018/10/30
INS002	CJ	BL711110	2018/10/30
INS002	CJ	BL161277	2018/10/30

The Lesson Quotation Tabsheet – general coding, select and distinct



For this tabsheet a quote is generated for a learner based on the Licence code, the number of lessons and the date on which the learners licence expires.



The code for the **btnQuoteOnClick** event handler to generates a quote for a number of drivers' lessons.

- Get the different Licence codes using table operations and populate the applicable combobox
- The number of Lessons and Learners Expiry date as input from the user are obtained , using a datetimepicker and a spinedit component.
- The licence codes that can be entered are: A,A1,EA,EB,C1,EC1,B,C,EC. The code must be categorized as A, B and C respectively to get a discount depending on the number of lessons.

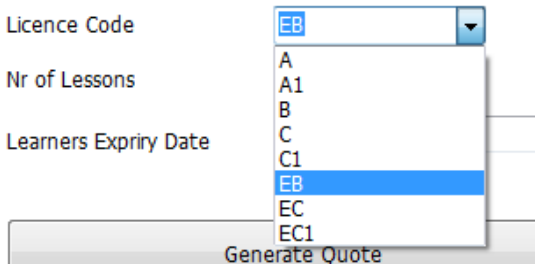
The percentage discount is calculated as follows:

Number of lessons	Category A	Category B	Category C
1-6	0.12	0.075	0
7-12	0.2	0.15	0.05
More than 12	0.25	0.2	0.1

- If the number of days left before the licence expires is less than 31 a 0.05 percent is added to the discount.
- The basic cost for a lesson is R375.00.

Generate the quote and display a message with the category code, the amount due and whether the learner client qualified for the discount.

When the tabsheet is shown the cmbLicenceCode combobox should be filled with all the applicable licence codes. The code to achieve this is shown next.




```
//-----------------------------------------------------
procedure TfrmQBDBExample.tbsGenerateLessonQuoteShow(Sender: TObject);
begin

 SQLQry := '';
  with dtmQBDB do
    begin
      qryQB2.Close;
      qryQB2.SQL.Text := ' Select distinct LearnerDriverCode From LearnerClients ';
      qryQB2.Open;
    end;
```

The name of the datamodule is dtmQBDB. We use the with clause to shorten our coding.

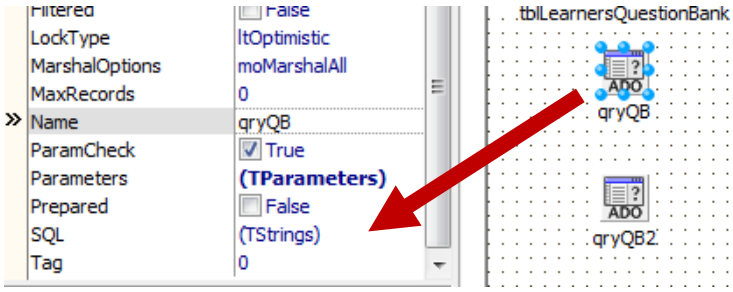
The query "dataset" is closed. We can now chage the corresponding query statemebnt

This query uses the distinct clause to select all the unique DriverLicenceCodes from the LearnerLicence

The Open method runs the query and the data is returned in the memory as part of the query dataset

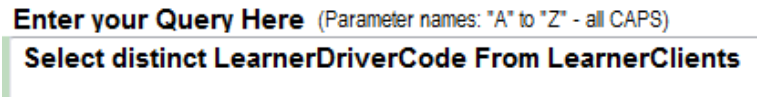
NOTE:The lines of code above is the typical code used to set an SQL statement and to "run the query" We can also treat the SQL statement as a set of items (Like that of a listbox) and do the following

```
      qryQB2.SQL.Clear;
      qryQB2.SQL.Add('Select distinct LearnerDriverCode From LearnerClients');
```



The SQL property of a query component is actually a collection of strings (TString Object, like a listbox)

When the following query is run



The query dataset (the qryQB2) in the memory will have the following records.

Table View | Query Result

LearnerDriverCode
A
A1
B
C
C1
EB
EC
EC1

The qryQB2 will have the following records in memory and the name of the applicable column/field is LearnerDriverCode

```
cmbLicenceCode.Items.Clear;

   with dtmQBDB do
    begin
      qryQB2.First;
      while not qryQB2.Eof do
        begin
             cmbLicenceCode.Items.Add(qryQB2['LearnerDriverCode']);
             qryQB2.Next;
        end;
    end;

 cmbLicenceCode.ItemIndex := 0;

 sedLessons.MinValue := 1;

 dtpLearnersExpire.DateTime := Now();
end;

//-----------------------------------------------------
```

Empty the combobox of any items

Goto the first record of the data in memory

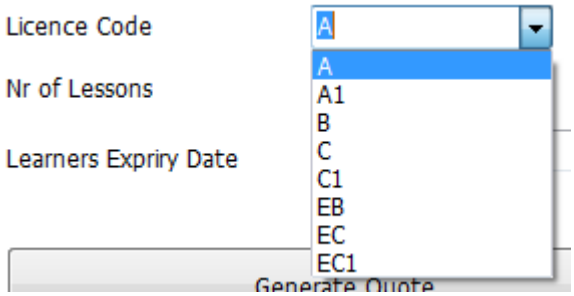
While we have not reached the last record yet

Take the value of the LearnerDriverCode colun of the query and add it to the combobox items

Goto the next record.

Show the first item in the combobox (A in our example)

Set the datetimepicker to todays date and time using the function Now()



The code for the onClick event [Generate Quote]

```
procedure TfrmQBDBExample.btnQuoteClick(Sender: TObject);
var
   dCode : char;
   sCode, sMessage : string;
   iLessons : Integer;
   rDiscPerc : Real;
   iDaysLeft : Integer;
   cQuotePrice, cDisCount : Currency;
   blnHurryDicount : Boolean;

begin
   sCode :=  cmbLicenceCode.Text;
   if (Pos('A',sCode) > 0) then  dCode := 'A';
   if (Pos('B',sCode) > 0) then  dCode := 'B';
   if (Pos('C',sCode) > 0) then  dCode := 'C';
```

Local variables declared. These variables can only be used as part of the method.

The Pos function is used to determine whether there is an A, B or C in the sCode.

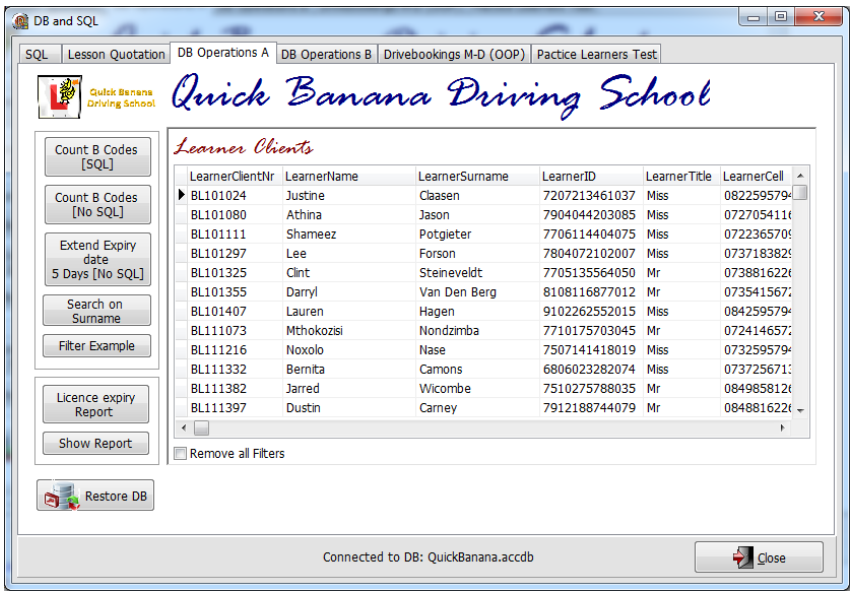
```
   blnHurryDicount := false;
   iLessons := sedLessons.Value;

   case iLessons of
     1..5:  begin
             if (dCode = 'A') then rDiscPerc := 0.12;
             if (dCode = 'B') then rDiscPerc := 0.075;
             if (dCode = 'C') then rDiscPerc := 0;
            end;
     6..12: begin
             if (dCode = 'A') then rDiscPerc := 0.2;
             if (dCode = 'B') then rDiscPerc := 0.15;
             if (dCode = 'C') then rDiscPerc := 0.05;
            end;
     else
            begin
             if (dCode = 'A') then rDiscPerc := 0.25;
             if (dCode = 'B') then rDiscPerc := 0.2;
             if (dCode = 'C') then rDiscPerc := 0.1;
            end;
   end;

   iDaysLeft := abs(DaysBetween(Now(),dtpLearnersExpire.DateTime));

   if (iDaysLeft < 31)  then
     begin
       rDiscPerc := rDiscPerc + 0.05;
       blnHurryDicount := true;
     end;

   cDisCount := 375 * sedLessons.Value * rDiscPerc;
   cQuotePrice := (375 * sedLessons.Value) - cDisCount;

   sMessage := 'For code [' + dCode + '] and ' + IntToStr(iLessons) + ' lessons ' + #13;
   sMessage := sMessage + 'You will need to pay: '
                        + CurrToStrF(cQuotePrice,ffCurrency,2) + #13;

   if blnHurryDicount then
     sMessage := sMessage + 'You Qualified for the HURRY-DISCOUNT!!';

   ShowMessage(sMessage);
end;
```

We use the Daysbetween function to return the number of days between to dates. The date from the current date and the date from the datetimepicker that the user as selected. .

To use the DaysBetween function we must add the DateUtils unit to our application, as part of the uses clause. For more functions available see:

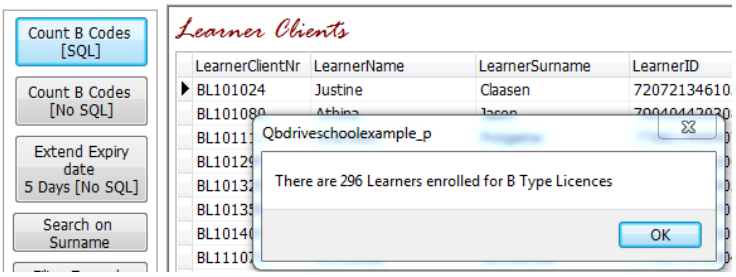
The DB Operations A Tabsheet

On this tabsheet different operations on the LearnerClients table are performed. Some using SQL statements and others implementing dataset methods.



The Count B codes SQL button – select and calculated field

When the user clicks on this button a messagebox is implemented to determine how many learners are enrolled for a B type drivers licence.



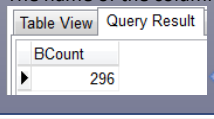
The code for the on click event of the button is presented next.

```
procedure TfrmQBDBExample.btnCountBSQLClick(Sender: TObject);
var
  BCounter : Integer;
begin
  SQLQry := '';
  with dtmQBDB do
    begin
      qryQB2.Close;
      qryQB2.SQL.Text := '';
      SQLQry := SQLQry + 'SELECT Count(*) as BCount ';
      SQLQry := SQLQry + 'FROM LearnerClients ';
      SQLQry := SQLQry + 'Where LearnerDriverCode = "B" ';
      qryQB2.SQL.Text := SQLQry;
      qryQB2.Open;
    end;
    BCounter := dtmQBDB.qryQB2['BCount'];

  ShowMessage('There are ' + IntToStr(BCounter) + ' Learners enrolled for B Type Licences');
```

The query is closed and the SQL statement: S

SELECT count(*) as BCount
FROM LearnerClients Where LearnerDriverCod

counts the records where the driver code is "B". The count function is an aggregate function and only one record is returned.
The name of the column is BCount



The value of the BCount field is assigned to the variable BCounter for display purposes

```
end;
```

The Count B codes SQL [No SQL] button and general code

The next button perfoms the same functionality as the previous button but using dataset methods instead of SQL.

```
procedure TfrmQBDBExample.btnCountBNOSQLClick(Sender: TObject);
var
  BCounter : Integer;
begin
  BCounter := 0;
with dtmQBDB do
  begin
    tblLearnerClients.First;
    while not tblLearnerClients.Eof do
      begin
         if (tblLearnerClients['LearnerDriverCode'] = 'B') then
           begin
               inc(BCounter);
           end;
       tblLearnerClients.Next;
      end;
  ShowMessage('There are ' + IntToStr(BCounter) + ' Learners enrolled for B Type Licences');

  end;
end;
```

This code iterates through all the records of the LearnersClient table and if the value of the LearnerDriverCode field = B then a counter is incremented.

Example (extra) – with filter and recordcount

In order to get the same answer, we could also follow an approach to limit the dataset to only display the "B" LearnerDriverCode records. We can do this by applying a filter to the records.

Note: Using a filter is like placing a WHERE condition on the data (records) that is in the dataset (table).

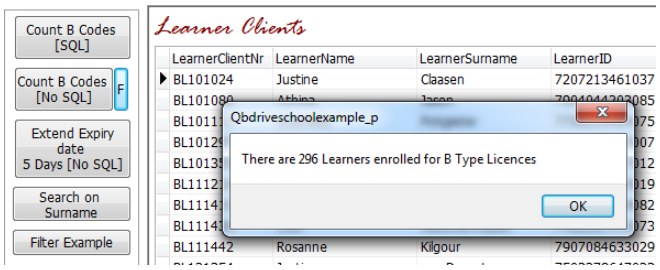
The Embarcadero wiki contains a very good explanation of how to use and apply filters. See: http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Setting_the_Filter_Property

```
procedure TfrmQBDBExample.btnFilterCountBNoSQLClick(Sender: TObject);
var
  BCounter : Integer;
begin

with dtmQBDB do
  begin
    tblLearnerClients.Filter := ' LearnerDriverCode = ' + QuotedStr('B');
    tblLearnerClients.Filtered := true;
    BCounter := tblLearnerClients.RecordCount;
    ShowMessage('There are ' + IntToStr(BCounter)
       + ' Learners enrolled for B Type Licences');

    tblLearnerClients.Filtered := false;
  end;
end
```

The first line sets a filter string to resemble: LearnerDriverCode = 'B'. This is assigned to the filter property of the table component. Setting the fltered property to active will then limited the records in the dataset to B drivercodes only.

If we now count the active records it will return the number of records with B Learnerdrivercodes.

Setting the filter property to false deactivates the filter and all the records of the table will be shown again.



Extend Expiry date 5 Days [No SQL]

We can also modify the data in a table (i.e. dataset) by using standard methods.

The following wiki has a nice description of the operations that can be applied to a dataset (i.e., table) http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Modifying_Data

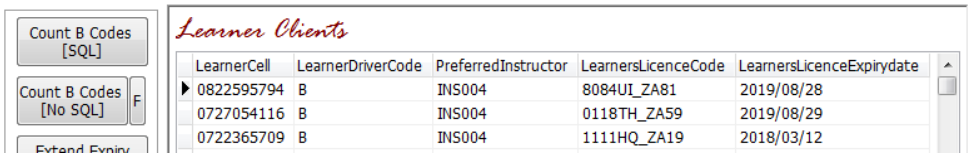
In this example we want to add 5 days (i.e. extend) the expiry date of all the learners drivers licences.

We need to iterate through all the records and for each record tell the DB engine that we are to modify (i.e. edit) the current record. We change the applicable fields and then move on to the next record.

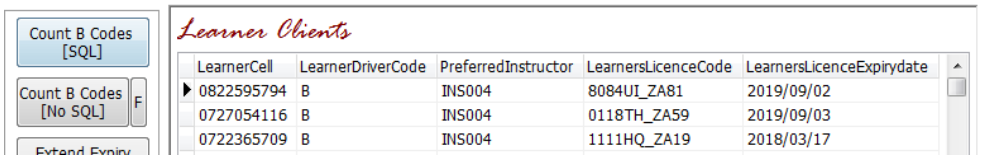
```
procedure TfrmQBDBExample.btnAddFiveDaysClick(Sender: TObject);
begin
   with dtmQBDB do
    begin
    tblLearnerClients.First;
      while not tblLearnerClients.Eof do
        begin
             tblLearnerClients.Edit;
             tblLearnerClients['LearnersLicenceExpirydate'] :=
                  tblLearnerClients['LearnersLicenceExpirydate'] + 5;
             tblLearnerClients.Next;
        end;
    end;
end;
```

This code iterates through all the records of the LearnersClient table and set each record state to dsEdit where we can modify a field (or more than one field value)

Some records before



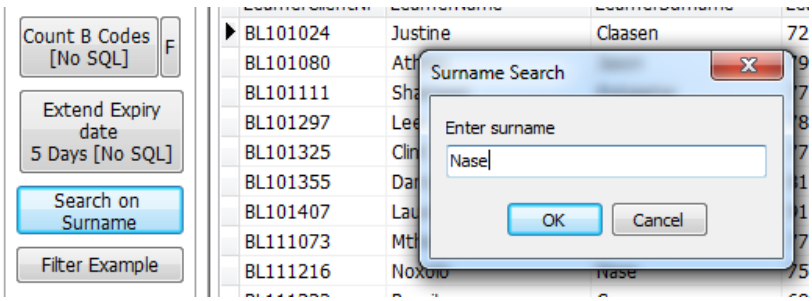
Some records after

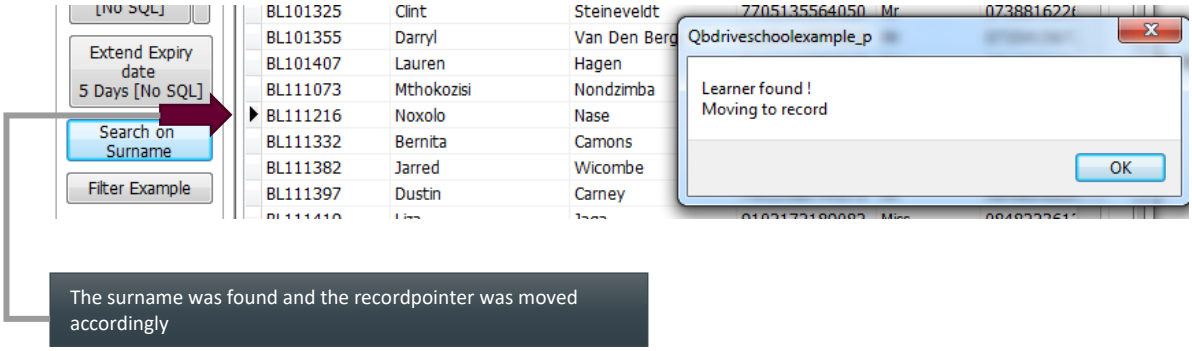


Search on Surname (and sorting a dataset)

We can use the LOCATE method to search and move the recordpointer to a particular record based on a search field (or fields). In this example the user enters the surname of a learner and if a matching record is found the record pointer is moved to that record.

Take note the LOCATE method moves the recordpointer to the matching record if such a record is found.





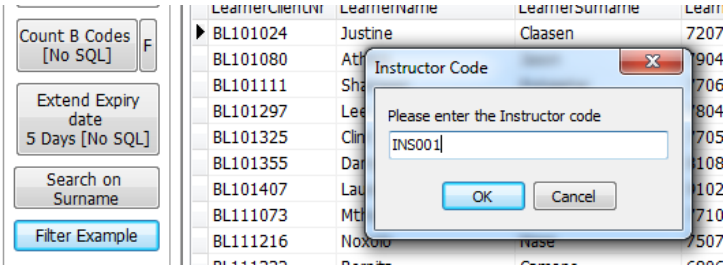
Code

```
procedure TfrmQBDBExample.btnSearchClick(Sender: TObject);
var
 sName : String;
begin
  sName := InputBox('Surname Search','Enter surname','');

  with dtmQBDB.tblLearnerClients do
   begin
    open;
    if Locate('LearnerSurname',sName,[])  then
      begin
        ShowMessage('Learner found !' + #10 + 'Moving to record');
      end
    else
      begin
        ShowMessage('Learner not found !');
      end;
   end;
end;
```

The Locate method takes three parameters:
1) The name of the field or fields that must be searched on
2) The value that must be searched for
3) Options relating to the applicability of case sensitivity [] results in ignoring the case

Filter Example (another)

In this example two dataset methods are applied. The sort method and the filter method.
The user enters the instructor code of an applicable instructor (i.e. the preferred instructor) and all learner records who have the applicable instructor code set as their preferred instructor is displayed in the dataset.



```
procedure TfrmQBDBExample.btnFilterClick(Sender: TObject);
begin

with dtmQBDB.tblLearnerClients do
  begin
    Active := true;
    filter := 'PreferredInstructor= '+ QuotedStr(InputBox('Instructor Code',
                                'Please enter the Instructor code','')); 
    sort := ' LearnersLicenceExpirydate ';
    filtered := true;
  end;

// given code
 SetGridColumnWidths(dbgrdLearnerClients);
end;
```

Filter is set to the instructor code and must be placed between ' ' eg. 'INS001' i.e. (qoutedstring)

Sort the records according to the expiry date (ascending is the default) if we add desc it would be sorted in descending order.

Activating the filter

4 Clients have INS001 as their preferredinstrutor. Only the filtered records are shown.

Learner Clients

LearnerClientNr	LearnerName	LearnerSurname	LearnerID	LearnerTitle	LearnerCell	LearnerDri
BL721008	Devin	Damons	7311116047011	Mr	0827054116	C1
BL891264	Hetal	Appavoo	7812212854073	Miss	0732595794	A
BL111332	Bernita	Camons	6806023282074	Miss	0737256713	C1
BL421272	Rory	Plaatjies	7801177525031	Mr	0737054116	EB

In order to remove the filter and show all the data (i.e. records) of the table again we must set the filter property to false.

Learner Clients

LearnerClientNr	LearnerName	LearnerSurname	LearnerID	LearnerTitle	LearnerCell	LearnerDri
BL411002	Damien	Ally	7711028735095	Mr	0824244936	B
BL261362	Darren	Van Laar	7611035684024	Mr	0844244936	EB
BL761349	Andrew	Tuck	7302237758017	Mr	0738816226	B
BL151197	Mandisa	Mapela	9206113978002	Miss	0725944470	B
BL721008	Devin	Damons	7311116047011	Mr	0827054116	C1
BL441100	Tamryn	Nicholls	8507284348007	Miss	0725179437	B
BL891109	Sizwe	Matiwane	9007135210050	Mr	0728816226	B
BL321077	Ndiphiwe	Mayola	7705156960077	Mr	0725944470	B
BL861014	Furqan	La Bercensie	7610236818094	Mr	0829602087	B
BL881451	Scharde	Le Roux	7604214175050	Miss	0845415672	B
BL801258	Zukiswa	Zuma	7212183882082	Miss	0733685999	B
BL721147	Mara	Twigg	7909252024018	Miss	0724386383	EC

☑ Remove all Filters

The onClick event of the Checkbox has the following code:

```
procedure TfrmQBDBExample.ckbRemoveFiltersClick(Sender: TObject);
begin
   dtmQBDB.tblLearnerClients.Filtered := not ckbRemoveFilters.Checked;
end;
```

If the checkbox is checked then all filters on the LearnerClientstable should be set to false.

Creating a simple text file report

Reports are output documents produced by processing data. This is often done from data which is contained in a database. In the following example we create a report to show the details of learners who's learners licence has expired.

License expiry Report

When the user clicks on this button we would like to create a report that looks as follows.

```
                    EXPIRED LEARNERS LICENCE REPORT
                    ===============================

    Client           Learner Details     DateExpired
    ------          ----------------     ----------

  BL101111              Potgieter, S      2018/03/17
  BL111419                   Jaga, L      2018/05/16
  BL111442                 Kilgour, R     2018/06/08
  BL131237              Ramaokane, S      2018/05/19
  BL131288               Rousseau, J      2018/05/03
  BL141345                 Daniels, C     2018/06/22
  BL141495                 Mancoba, T     2018/05/30
  BL151016                  Rulsur, J     2018/06/08
```

10 Characters | 25 Characters | 15 Characters

```
  BL861014            La Bercensie, F     2018/03/11
  BL861028                 Burrell, R     2018/05/10
  BL871039              Lottering, T      2018/06/03
  BL881451                Le Roux, S      2018/03/11
  BL891023                Chowhan, E      2018/03/19
  BL891060                 Toyiya, M      2018/05/14
  BL891109               Matiwane, S      2018/03/08

                Total Expired Licences:       89
```

Let's examine the code used

```
procedure TfrmQBDBExample.btnReportClick(Sender: TObject);
var
   ClientID , SurnameInit, sDateExpired : String;
   iTotal : Integer;
   oRepFile : Textfile;
begin
   AssignFile(oRepFile,'LearnersExpiredReport.txt');
   Rewrite(oRepFile);

   iTotal := 0;
   Writeln(oRepFile,'EXPIRED LEARNERS LICENCE REPORT':50);
   Writeln(oRepFile,'===============================':50);
   Writeln(oRepFile,'');
   Writeln(oRepFile,'Client':10,'Learner Details':25,'DateExpired':15);
   Writeln(oRepFile,'------':10,'----------------':25,'----------':15);
   Writeln(oRepFile,'');

   with dtmQBDB do
    begin
     tblLearnerClients.First;
       while not tblLearnerClients.Eof do

         begin
```

Defining a textfile variable named oRepFile

Creating a reference to the file on disk, giving the file a name and, rewriting the content of the file if the file exists. If we wanted to add data to the end of the file and not delete any content then we would use the **Append** procedure

Using the writeln procedure to write data (i.e. text) to the textfile. The : indicates the column width.

We need to iterate through all the records and check wheter the learner licence has indeed expired by comparing the expiry date to todays date

Comparing the date

```
        if (tblLearnerClients['LearnersLicenceExpiryDate'] < Now()) then
          begin
            ClientID := tblLearnerClients['LearnerClientNr'];
            SurnameInit :=  Trim(tblLearnerClients['LearnerSurname']) + ', '
                          + copy(tblLearnerClients['LearnerName'],1,1);
            sDateExpired :=  DateToStr(tblLearnerClients['LearnersLicenceExpiryDate']);

            inc(iTotal);
            Writeln(oRepFile,ClientID:10,SurnameInit:25,sDateExpired:15);
          end;
        tblLearnerClients.Next;
      end;
    end;
   Writeln(oRepFile,'');
   Writeln(oRepFile,'Total Expired Licences:':40,iTotal:8);
   CloseFile(oRepFile);

end;
```

Extracting field values to variables

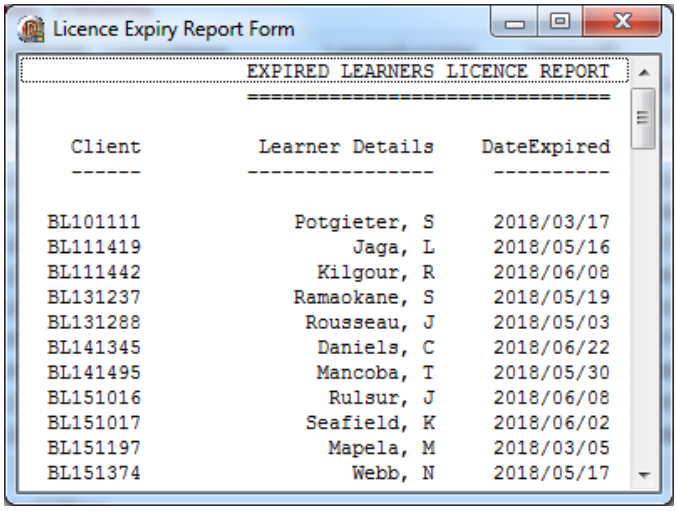
Writing the variables to the

Writing the counter variable

Closing the file will save it.

Show Report

In the on click event of this button we apply some clever dynamic instantiation to create a form with a listbox on it, to load the textfile for display purposes. The form is not part of the application and is created when the user clicks on the button.



Lets examine the code

```
procedure TfrmQBDBExample.btnShowReportClick(Sender: TObject);
var
  TempForm : TForm;
  TempListBox : TListBox;
begin
  TempForm := TForm.Create(frmQBDBExample);
  TempForm.Width := 400;
  TempForm.Height := 300;
  TempForm.Caption := 'Licence Expiry Report Form';
  TempForm.Position := poDesktopCenter;

  TempListBox := TListBox.Create(TempForm);
  TempListBox.Parent := TempForm;
  TempListBox.Align := alClient;
  TempListBox.Font.Name := 'Courier New';
  TempListBox.Font.Size := 9;
  if FileExists('LearnersExpiredReport.txt') then
```

Declaring a form and a listbox object variable

We instantiate the Form by calling the create constructor and stating that the form belongs to the main form of our application

Form must be displayed in the middle of our desktop

Only after the form as been created then the listbox may be instantiated and be part of the Tempform. The parent is the component on which the created object should be placed on.

The align alClient stretches the listbox over the canvas of the form

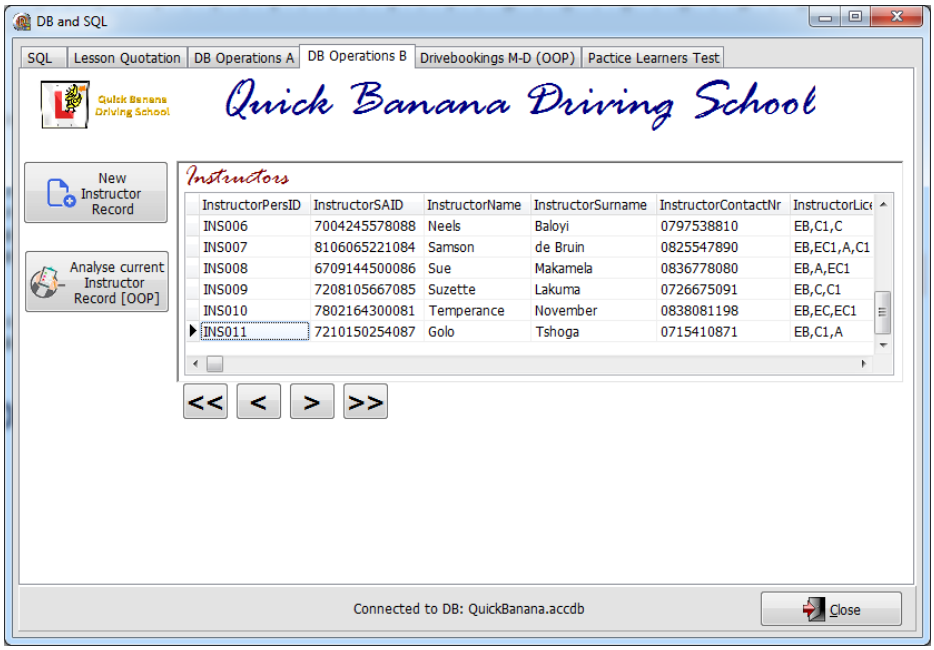
Loading the textfile if it exists into the listbox

Using the showmodal method to display the form

```
    begin
     TempListBox.Items.LoadFromFile('LearnersExpiredReport.txt');
     TempForm.ShowModal;
    end
  else
     MessageDlg('Report file does not exit. Please generate', mtError,
      [mbOk], 0, mbOk);
end;
```

The DB Operations B Tabsheet

As part of this tabsheet the user may browse the records of the instructors and insert a new instructor record to the table.



Browsing the instructor table records

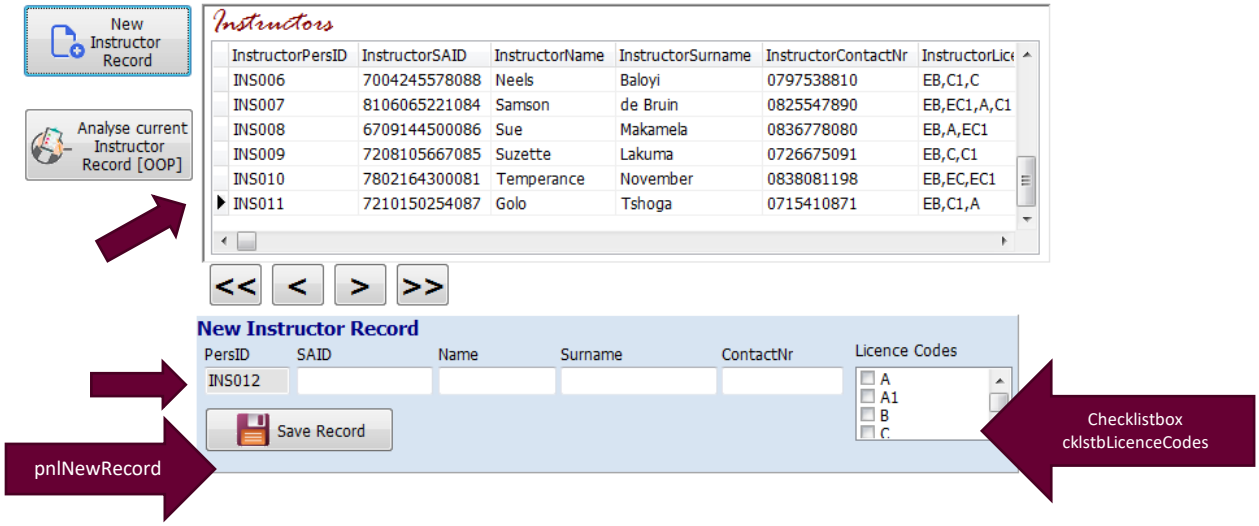
The following methods are used to move the recordpointer

```
procedure TfrmQBDBExample.btnFirstClick(Sender: TObject);
begin
   dtmQBDB.tblInstructors.First;
end;

procedure TfrmQBDBExample.btnPreviousClick(Sender: TObject);
begin
   dtmQBDB.tblInstructors.Prior;
end;

procedure TfrmQBDBExample.btnNextClick(Sender: TObject);
begin
   dtmQBDB.tblInstructors.Next;
end;

procedure TfrmQBDBExample.btnLastClick(Sender: TObject);
begin
   dtmQBDB.tblInstructors.Last;
end;
```

New instructor record button

When the user clicks on this button then he/she is allowed to add the data of a new instructor record to be inserted to the table.

A panel is shown with various components, which is used to capture the data of the new instructor.



You will notice that the last instructor record (i.e. Tshoga) has an instructor personel ID of INS011. The new instructor will there a personel ID of INS012. The different licence codes has also been added to an applicable checklistbox.

Lets examine the code of the onClick event of the [New Instructor Record] button.

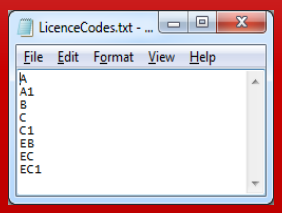
```
procedure TfrmQBDBExample.bttNewRecordClick(Sender: TObject);
var
  sInstID : String;
  LastVal : Integer;
begin
   pnlNewRecord.Visible := true;

   // Generate new Instructor ID
   dtmQBDB.tblInstructors.Sort := 'InstructorPersID';
   dtmQBDB.tblInstructors.Last;

   sInstID := dtmQBDB.tblInstructors['InstructorPersID'];
   Delete(sInstID,1,3); // Removing INS
   LastVal := StrToInt(sInstID);
   inc(LastVal); // adding 1 to lastval
   sInstID := IntToStr(LastVal);
   insert('INS',sInstID,1);
   while Length(sInstID) < 6 do
     insert('0',sInstID,4);
   lbledtPersID.Text := sInstID;

   // Loading data into checklistbox

   cklstbLicenceCodes.Items.Clear;
   cklstbLicenceCodes.Items.LoadFromFile('LicenceCodes.txt');

end;
```

Showing the panel

Ensuring that the instructorstable are sorted according to the InstrctorPersID, so that the highest value PersId is the last record.
TAKE note: Tables are sorted by default according to the PK index.

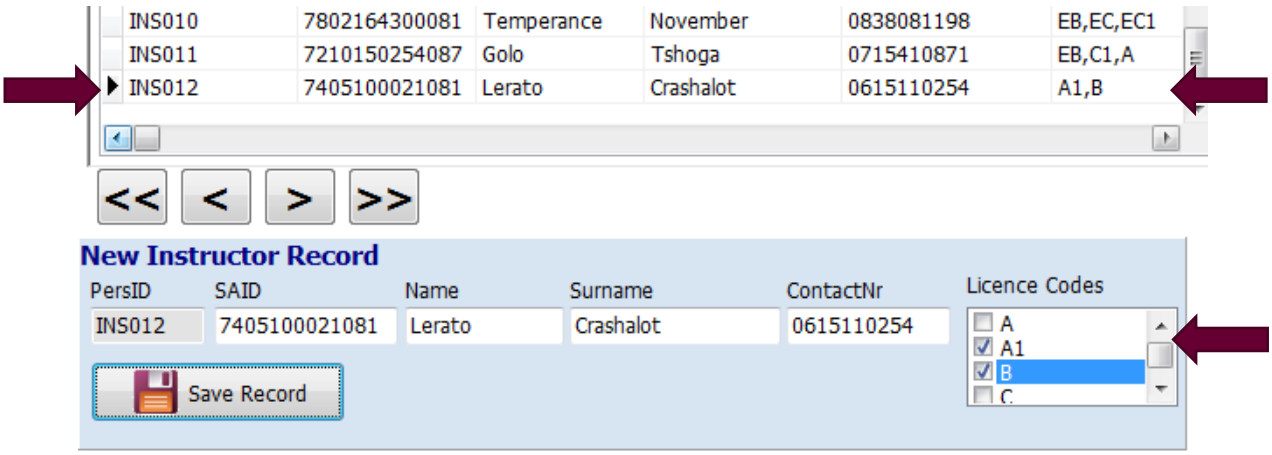
Obtaining the last record value, Eg. INS011. Removing the INS part which will result in 011, changing the value to an integer and adding 1 which wil result in 12. Converting 12 to a string, adding 0s to the front until the length is 3 characters long. Appending INS to the front INS012, assigning it to the applicable editbox.

The textfile licencecodes contain the different values of all the applicable codes that is offered. These are loaded as items to the checklistbox.



Save record button

When the user clicks on the Save Record button then the record is inserted into the table.



Let's examine the applicable code

```
procedure TfrmQBDBExample.bttSaveRecordClick(Sender: TObject);
var
  I : Integer;
  sCodes : String;
begin
  sCodes := '';
  with dtmQBDB do
    begin
      tblInstructors.Insert;
      tblInstructors['InstructorPersID'] := lbledtPersID.Text;
      tblInstructors['InstructorSAID'] := lbledtSAID.Text;
      tblInstructors['InstructorName'] := lbledtInstName.Text;
      tblInstructors['InstructorSurname'] := lbledtInstSurname.Text;
      tblInstructors['InstructorContactNr'] := lbledtContactNr.Text;

      for I := 0 to cklstbLicenceCodes.Count - 1 do
        begin
          if (cklstbLicenceCodes.State[I] = cbChecked) then
            begin
              sCodes := sCodes + cklstbLicenceCodes.Items.Strings[I] + ',';
            end;
        end;

     Delete(sCodes,Length(sCodes),1); // delete the last , appended

     tblInstructors['InstructorLicenceCodes'] := sCodes;
     tblInstructors['EmpBasicSalPA'] := 125000;
     tblInstructors.Post;

    end;
end;
```

The Table (dataset) is placed into insert mode which opens as new row.

The data entered in the various editboxes are assigned to their corresponding fields

We need to iterate through all the items in the checklistbox and create a string that contains the data of the checked codes only

The starting salary is 12500 PA

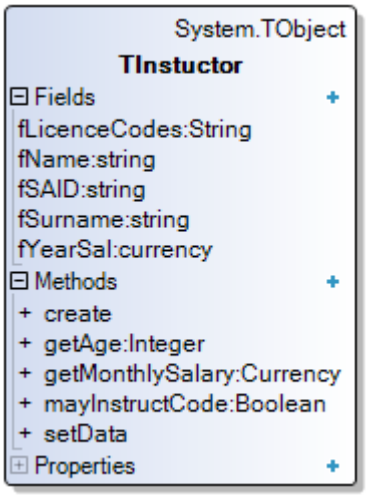
We save the record to the table

You may add code to clear all the edits and hide the panel again, after the record as been saved

The Analyze Current Instructor Record [OOP]

Two user defined classes have been incorporated as part of the complete example application. For this button one class object is implemented through the calling of certain methods to analyse the data of the current instructor record.

The applicable class diagram is presented below.



The method ***setData*** receives 5 parameters and sets the applicable class fields accordingly

```
procedure TInstuctor.setData(fN : String; fSur : String ; fID : String; fLicCodes : String;
fYSal : currency );
  begin
    fName := fN;
    fSurname :=  fSur;
    fSAID := fID;
    fLicenceCodes :=  fLicCodes;
    fYearSal := fYSal;
  end
```

The ***getAge*** method uses code to determine the age of the instructor based on his/her ID

```
// E.g. ID 9504295018081
function TInstuctor.getAge:Integer;
 var
   YY , DD , MM, sBDate : String;
   bDate : TDate;
  begin
     YY := copy(fSAID,1,2);
     MM := copy(fSAID,3,2);
     DD := copy(fSAID,5,2);

     if (YY >= copy(DateToStr(bDate),1,2)) then
       sBDate := '19'+YY+'/'+MM+'/'+DD
     else
       sBDate := '20'+YY+'/'+MM+'/'+DD;

     bDate := StrToDate(sBDate);
     Result := YearsBetween(Now(),bDate);
   end;
```

The ***getMonthlySalary*** method returns the monthly gross salary of the instructor

```
function TInstuctor.getMonthlySalary:Currency;
  begin
    result := fYearSal / 12;
  end;
```

When the user clicks on the [Analyse Current Instructor Record] button then the data of the current record, i.e. where the recordpointer points to is sent as parameters to an instantiated object named TempInstructor.

A message string is created by calling two methods and displaying it to the user.

```
procedure TfrmQBDBExample.bttAnalyzeCurrentRecord(Sender: TObject);
var
   TempInstructor : TInstuctor;
   sMessage : String;
begin
   TempInstructor := TInstuctor.create;

   TempInstructor.setData(dtmQBDB.tblInstructors['InstructorName'],
   dtmQBDB.tblInstructors['InstructorSurname'],
   dtmQBDB.tblInstructors['InstructorSAID'],
   dtmQBDB.tblInstructors['InstructorLicenceCodes'],
   dtmQBDB.tblInstructors['EmpBasicSalPA']);

   sMessage :='Instructor ' + dtmQBDB.tblInstructors['InstructorSurname'] + #13;
   sMessage := sMessage + 'is ' + IntToStr(TempInstructor.getAge) + ' years old '+ #13;
   sMessage := sMessage + 'and earns a monthly salary of ';
   sMessage := sMessage + CurrToStrF(TempInstructor.getMonthlySalary,ffCurrency,2);

   ShowMessage(sMessage);

end;
```

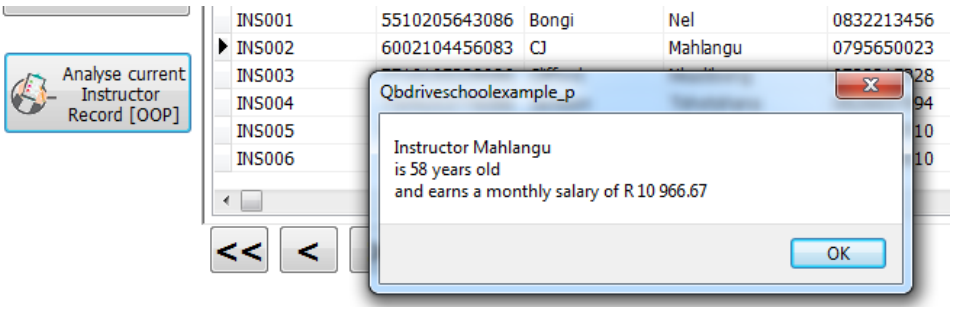
TempInstructor opbject declared, of the type TInstructor

TempInstructor opbject instantiated by calling the constructor

Invoking (calling) the setData method and passing all the applicable fields as parameters.

Constructing a message string containing the result of two of the methods of the class as well

In the example below the record pointer (current record) refers to the record of CJ Mahlangu (INS002).



The Drivebookings MD Tabsheet

This tabsheet contain some additional functionality.

Another backend class (user defined class is implemented) The applicable class object is used with the user clicks on the [Analyse Lesson Booking Records for Instructor] button.

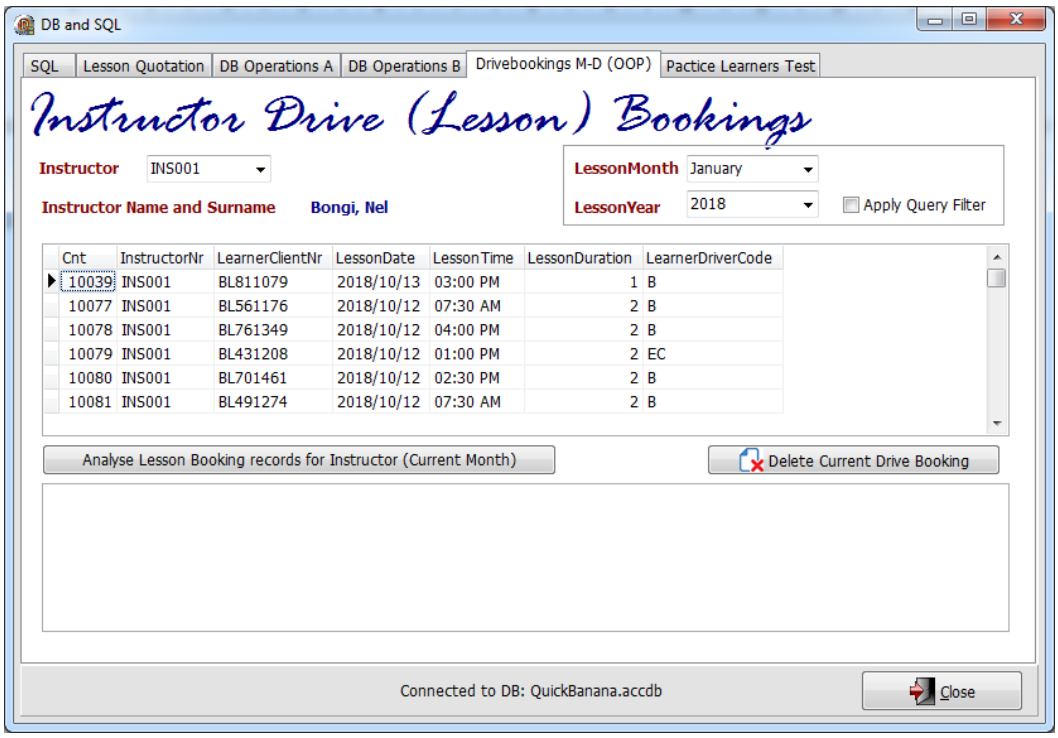
The data displayed in the applicable DB grid is based on a SQL query which utilises various where conditions with parameters set to “filter” the applicable data. The SQL query implements a join on two tables.

The user is allowed to select an instructor using the combobox. When an instructor is selected then all the lessons booked for that applicable instructor is displayed, in the DBGrid. The user may further refine the query by selecting the LessonMonth and the LessonYear if that is done then only the lessons booked for that particular month and year for the particular instructor is shown. In order to apply the selection the user must also select and click the [Apply Query Filter] checkbox. De-checking the checkbox discards the month and year criteria and all the lessons for the instructor is shown again.

The Analyse Lesson Booking Records for Instructor (Current Month)

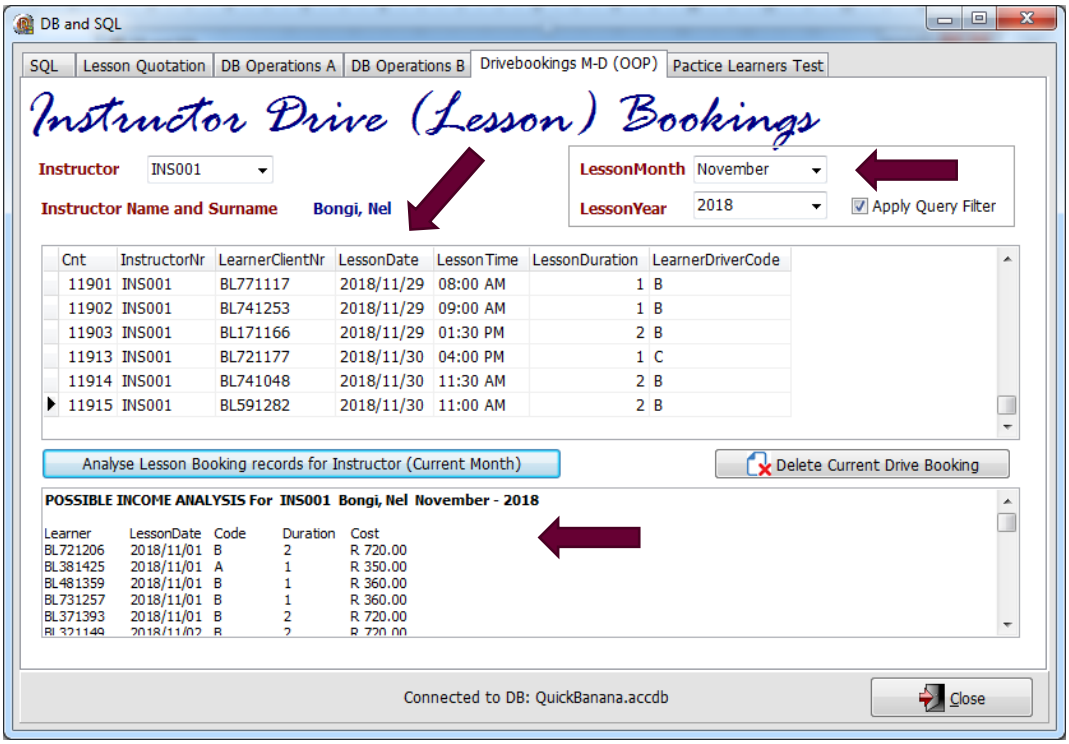
This button implements a back end class to present a report on the possible income for the instructor for the current month.

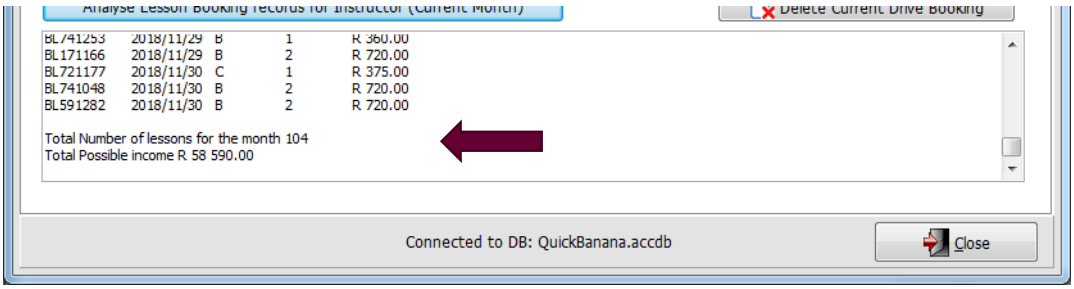
We will not look at the code for this functionality. You are welcome to examine the given code and inspect how when the tabsheet is shown the initial data is set up for display and how the varions components like the comboboxes are populated.



The class TLessonCost is used to determine the cost of the lesson for the current booking.

NOTE: Please also look at this class and how it is used in the application. The applicable class object is used to do an analysis of the lessons and present the output as indicated in the screenshots below.

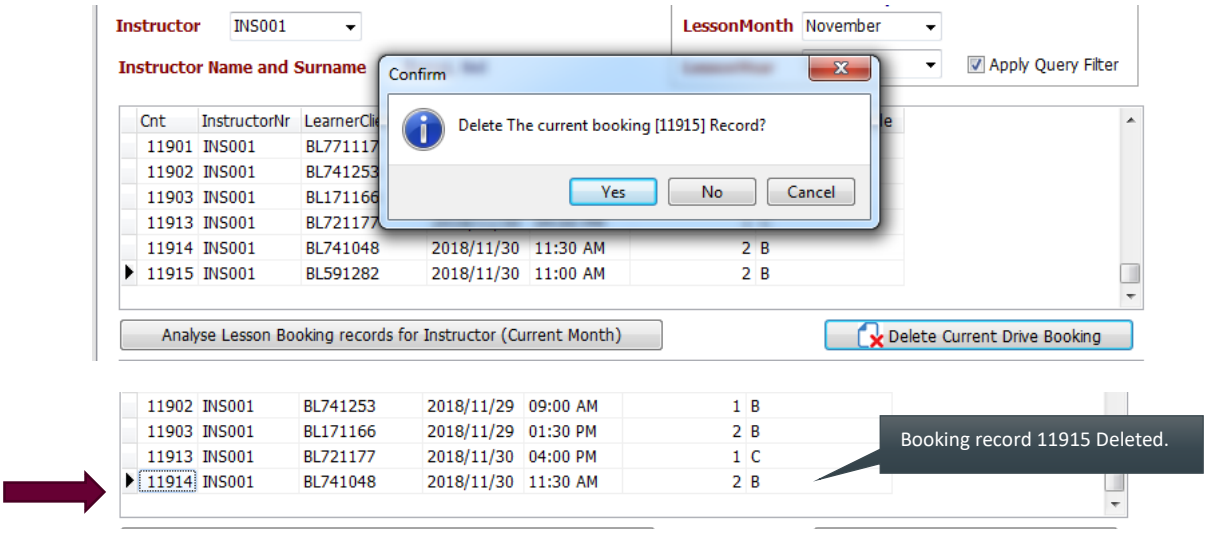




The Delete Current Drive booking button

When the user clicks on this button the current drive booking record from the corresponding bookings table is deleted, after the user confirms his/her choice.

We will look at the applicable code



In order to understand the code below we must fist understand that the data displayed in the DB grid is based on a query. Therefore we must fist obtain the current booking cnt field value from the query, then search (locate) the corresponding booking record in the Drivebookings table where we can then delete it calling the Delete method.

We call the cmbInstructorCodeChange method to refresh the data as if the user selected a new instructor.

```
procedure TfrmQBDBExample.bttDeleteDrivebookingRecordClick(Sender: TObject);
begin
  // We must first locate the current corresponding record
  // in the table as we are working with a query

    if MessageDlg('Delete The current booking ['
                 + IntToStr(dtmQBDB.qryQB2['Cnt'])
                 + '] Record?', mtConfirmation, mbYesNoCancel, 0) = mrYes then
      begin
      dtmQBDB.tblDriveBookings.Open;
       if dtmQBDB.tblDriveBookings.Locate('Cnt',dtmQBDB.qryQB2['Cnt'],[]) then
         begin
            dtmQBDB.tblDriveBookings.Delete;
            cmbInstructorCodeChange(self);
         end;
      end
     else
     begin
       Abort;
     end;
end;
```

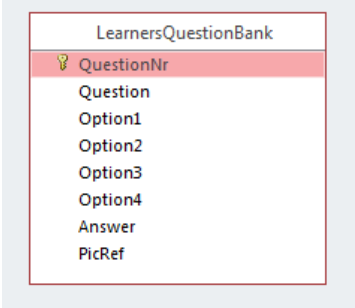
Locating the corresponding record in the table

Deleting the corresponding record

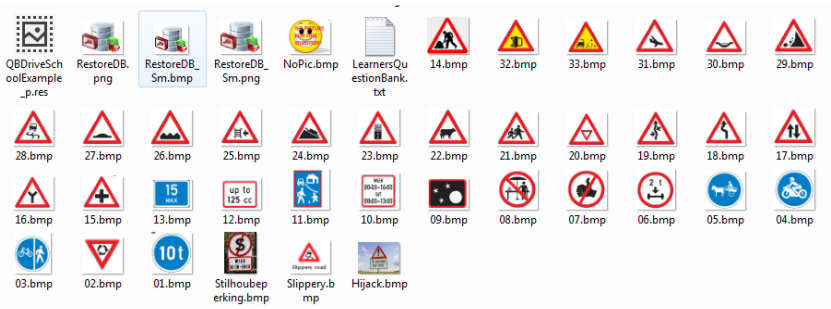
"Refreshing the tabsheet"

The Practice learners test Tabsheet

The operations of this tabsheet uses data from the LearnersQuestionBank table of the database.



The images for the questions is stored in the application folder.



The following public form class variables are used as part of the logical implementation of the code

```
  // Learner test public declarations

  arrQuestionAskedAlready : Array [1..46] of boolean;
  Answer : integer;
  QuestionsAskedCount : Integer;
  CorrectCount : Integer;
end;
```

The array represents a “parallel” array to the question records in the table and is used as a flag array to indicate whether a question has already been asked or not.
We are not going to discuss the logic and the code for this tabsheet. You are more than welcome to examine the code and follow the logic used as part of the applicable methods and event handlers implemented.